nigeria computer society

www.ncs-nig.org

CONFERENCE PROCEEDINGS

Volume 16

8th International Conference

INFORMATION TECHNOLOGY CAPACITY BUILDING:

The future of Nigeria's Economc Growth (ICABUILD 2005)

Edited by: AJIBIKE O. ITEGBOJE

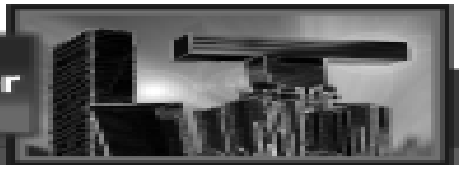
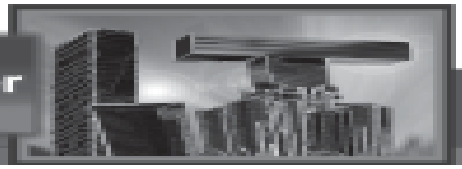


TABLE OF CONTENTS

IT CURRICULUM RESTRUCTURING

OUTSOURCING IMPLICATIONS FOR IT CAPACITY BUILDING



An Appraisal of Information Technology (IT) Curriculum in Nigeria Tertiary Institutions

Sadiq, F.I., Adetunmbi, A. O; Oludapo, A.,

ABSTRACT

Knowing the importance and the relevance of Information Technology, (IT) in our Society with its current level of development in the nation at large, there is the need for continuous awareness and adequate training of IT in all levels of education right from Secondary School Stages. This is to ensure high level of practical performance in the IT industry in order to enhance manpower development in the country, which can be achieved by proper designing of IT curriculum. This paper enumerates the various ways by which IT curriculum can be improved on from Secondary School to Tertiary Institutions and also highlights the benefits that can be derived from such restructing.

1. INTRODUCTION

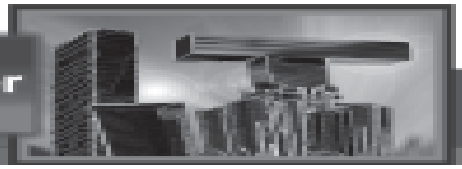
According to Oxford English dictionary a curriculum, is defined as a course of study (Oxford, 1999). This implies that curriculum is the first stages or things to be considered before any school or tertiary institutions became operational for whatever programme they intend to run, for the purpose of training people for manpower development. Curriculum as stated above is also very important, in that it serves as a guide for teachers, tutors, technologist and even lecturers in whatever course they are teaching or lecturing in their respective schools and institutions which varies from one department to another, for instance the curriculum of Mathematics, Zoology, Biochemistry are completely different from that of Computer Science degrees programme. In other words, in this paper our concern is basically on the curriculum of Information Technology (IT), if any and how the existing ones can be restructured to meet up with the current challenges (FNSAAU, 2004).

On the other hand, IT is the technology involved in Acquiring, Storing, Processing, and Distribution of Information by electronic means (including radio, television, telephone, computer etc). It is therefore the product of the marriage between Computer Technology (essentially for information acquisition, storage and processing) and Telecommunication Technology, which is for information distribution. Hence Computer +Telecommunications = IT is generally known as Information Communication Technologies (ICTs) (Falaki, 2002).

In recent years, a mixture of communication technologies has been occurring, serving to continually extend the capabilities of communication networks. The scale of this movement suggests that many of the central issues for education in the coming years will be driven by technological change. Indeed, the influence of these new technologies is such that they may well form an underlying mechanism for the future of education. From the perspective of curriculum development there is need to have a unified structure curriculum for IT courses in Nigeria Institution right from Secondary School with a foundational course introduce from Primary Schools so as to create new challenges (Hardin, 1998).

2. TECHNOLOGICAL CHANGE AND CURRICULUM

Technological change is redefining not only how we communicate, but in turn, is redefining how we need to educate. The ready availability of information has lessened the necessity of learning, but raises new issues in terms of effective searching and the development of an ability to evaluate information. The development of systematic skills and higher order thinking is increasingly an important focus. The stakeholders and interest groups in this process are many and varied, with pressure for change and reform brought from teachers, schools and school councils, government authorities, industry and students themselves. All have differing perspectives on the best curriculum planning approach



to deal with this change.

The root of change is increasing in technological and networking capacity, but what are the ramifications for curriculum development, and interpretations necessary for the classroom? Many educators hold the view that computers and Internet connectivity are "tools" for learning and thus, believe that an increased grade point average is often the only measure of value for technological resources. However, an important perspective for all educators, government administrators and school boards to consider is, that networking represents not just a new set of tools, but a new environment for learning and teaching. New communications technologies encourage new possibilities and develop new requirements. An outcome based education policy must accept that the new communications technologies must impact upon core curriculum requirements and will influence diversity, context, assessment issues and practice.

3. TECHNOLOGY RESOURCES

Kennedy states that, "...curriculum developers must reflect on actual practice to understand ...(appropriate)...curriculum development practice" There is a debate about the cost / value relationship of technology implementation in secondary education. While much technology may provide long-term advantages in educational budgeting, provided resources are allocated towards efficient collaboration. This is, however, dependent on a curriculum focus which identifies the need for collaboration and supports its' implementation (Brandy and Kennedy, 1999).

However, as a symbol for recent development and dynamism some Secondary Schools already claimed that they have Curriculum for IT, which means they have started teaching basic few foundational courses for Information Technology. The Schools under this category are Owajoba Secondary School, Akure, Ondo State. Ofua Memorial Secondary School, Uromi, Edo State. Salawu Abiola Comphrensive High School, Abeokuta, Ogun State etc. Though, from our findings it was observed that what they have in their curriculum are basic computer science courses and not IT parse. But in any case they are better than secondary school that don't teach computer science at all.

However, we all know that IT evolved as components of Computing Technology, which also comprises of Computer Science, Electronic Technology, Information system etc. Although, what we are aware of by now is that there exist a standard curriculum for all tertiary institutions in Nigeria for Computer Sciences and even of recent, there was a workshop by NUC held at the Department of Computer Science, University of Ibadan. Where all Computer Science departments from the whole Nigeria Institutions were invited. In the workshop the existing curriculum for Computer Science undergraduate programme for both Conventional Universities and University of Technologies were deliberated upon and later reviewed in that same gathering. Below are few out of the approved courses in the current curriculum:

From our survey we studied the curriculum of the following institutions namely; Ambrose Alli University, Ekpoma. Edo State, Federal University of Technology, Akure. Ondo State, University of Benin, Edo State etc. Although many people claimed that one of the private institution by name Covenant University, Otta, Ogun State already have IT curriculum in place, if this argument is true then all private institutions may be using the same curriculum for all there programmes. But, from our findings the said IT curriculum for the university is not far from the existing ones for computer sciences in other tertiary institutions, this can be seen clearly from the table 3 presented above. Hence their courses are devoid of core IT curriculum as seen in universities outside this countries: such as Canada, USA, Carolina, etc. Hence, the objectives of the said institution is tailored towards IT manpower development and are listed as follows:

To develop manpower with skills and knowledge needed to meet the requirements of a rapidly advancing and

100 Level		
Courses	**Title**	**Units**
CSC 101	Introduction to Computer Science	3
CSC102	Introduction to Computing	3
CSC 132	Introduction to Compiler Construction	3
MTH 101	General Maths I	3
LIB 101	Library Skills	1
200 Level		
CSC 211	Logic Design	3
CSC 212	Computer Hardware	3
CSC 231	Structured Programming	3
CSC 233	Object Oriented Programming	3
300 Level		
CSC 302	System Analysis and Design	3
CSC 304	Computer Networks	3
EPS 201	Entrepreneurship Study I	2
CSC 399	Industrial Training II	3

Table 3.: Above from 100 to 400 Levels present the curriculum for undergraduate B.Sc. Computer science (4yrs) and B.Sc. Management Information System (4yrs).

400 Level		
Courses	**Title**	**Units**
CSC 412	Computer Architecture	3
CSC 421	Computer Operating System II	3
CSC 423	Web Design and Data Security	3
CSC 431	Data Communications	3
CSC 433	Software Engineering II	3
CSC 492	Selected topics in Computer Science	6
CSC 495	Projects	6

Table 2: *Continuation of the Current Curriculum for Computer Science Courses.*

(Source: NUC workshop 2004, University of Ibadan).

Table 3: (Course Curriculum source Covenant University)

100 Level

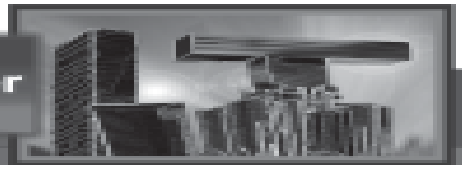
Course Code	Course Title	Course Pre-Requisite	Credit Unit	L -T -P. in Hrs	Semester
PHY112	Physics I	-	2	22-8-0	α
PHY113	Physics II	-	2	22-8-0	α
MAT 112	Mathematics I	-	2	22-8-0	α
MAT113	Mathematics II	-	2	22-8-0	α
PHY119	Physics Laboratory I	-	1	0-0-45	α
CIS111	Introduction to Computer Science	-	2	30-0-0	α
CHM111	Chemistry I	-	3	35-10-0	α
CHM119	Chemistry Laboratory I	-	1	0-0-45	α
PHY122	Physics III	-	2	22-8-0	?
PHY123	Physics IV	-	2	22-8-0	?
MAT123	Mathematics III	-	2	22-8-0	?
MAT124	Mathematics IV	-	2	22-8-0	?
PHY129	Physics Laboratory II	-	1	0-0-45	?
CIS 121	Computer Programming I	-	2	15-0-15	?
CHM121	Chemistry II	-	3	35-10-0	?
CHM129	Chemistry Laboratory II	-	1	0-0-45	?
CST111	Computer Application I	-	1	10-0- 15	α
EDS111	Entrepreneurial Development Studies I	-	1	15-0-0	α
TMC111	Total Man Concept I	-	1	15-0-0	α
TMC112	Physical Health Education I	-	0	0-0-45	α
CST121	Computer Application II	-	2	15-0-15	?
EDS121	Entrepreneurial Development Studies II	-	1	15-0-0	?
TMC121	Total Man Concept II	-	1	15-0-0	?
TMC122	Physical Health Education II		0	0-0-45	?
GST111	Use of English I	-	2	30-0-0	α
GST121	Use of English II	-	2	30-0-0	?

200 Level

Course Code	Course Title	Course Pre-Requisite	Credit Unit	L -T -P. in Hrs	Semester
CSC211	Computer Programming II	CIS111, 121	3	35-10-0	α
CSM211	Mathematical Methods I	CST114	3	35-10-0	α
CSC212	Computing Laboratory I	-	1	0-0-45	α
CSM212	Statistical Methods	-	2	22-8-0	α
ECO111	Principles of Economics	-	3	35-10-0	α
ACC111	Introduction to Accounting	-	3	35-10-0	α
CSC221	Object Oriented Design & Programming in C ++	CIS 111, 121	3	35-10- 0	?
CSC222	Computing Laboratory II	-	1	0-0-45	?
CSC223	Numerical Computation	-	2	22-8-0	?
MIS221	Introduction to Management Information System	-	2	22-8-0	?
MIS222	Principles of Management	-	3	35-10-0	?
CSM221	Mathematical Methods II	CST 124	2	22-8-0	?
CSM229	SIWES: Industrial Training I	-	3	0-0-135	Post-? 3 month holiday

300 Level

Course Code	Course Title	Course Pre-Requisite	Credit Unit	L -T -P. in Hrs	Semester
CSC311	Discrete Mathematics	-	3	35-10-0	α
CSC312	Data Structures & Algorithm	-	3	35-10-0	α
CSC313	Computer Programming IV	CSC222	2	22-8-0	α
CSC314	Theory of Computing	-	2	22-8-0	α
CSC315	Computer Organization & Programming	-	3	35-10-0	α
CSC321	Computer Graphics	CSM223	2	22-8- 0	?
CSC322	Database Design & Management	-	2	22-8-0	?
CSC323	Logic Programming	CSC212	3	35-10- 0	?
CSC324	Compiler Design		2	22-8- 0	?
CSC325	Operating Systems		3	38-7-0	?
CSM321	Information System Analysis & Design	-	2	22-8- 0	?
EDS311	Entrepreneurial Development Studies V	EDS211, 221	1		15-0-0



400 Level

CSP411	Computer Animation	-	2	15-0-15	α
CSP412	Introduction to Bioinformatics	-	2	30-0-0	α
CSP413	Introduction to natural language processing	-	2	30-0-0	α
CSP421	System Administration	-	2	30-0-0	?
CSP422	Fuzzy Logic	-	2	30-0-0	?
CSP423	Object Oriented Design	CSC222	2	30-0-0	?
CSP424	Software Development Techniques	-	2	30-0-0	?
CSP425	Virtual Reality	-	2	30-0-0	?
EDS411	Entrepreneurial Development Studies VII	EDS311, 321	1	15-0-0	α
TMC411	Total Man Concept VII	TMS311, 321	1	15-0-0	α
CSC424	Neural Network		2	30-0-0	?

challenging field of IT and Management. To produce graduates with IT skills and prepare them for global competitiveness. To produce managers with the spirit of self-reliance and encourage them to adopt holistic approach in professional pursuit. Nearly all the Universities mentioned as a focus on the aforementioned objectives.

However, there is a serious need to restructure the above curriculum or have separate department whose curriculum will outlined or feature IT and application proper in their curriculum. Above all:

IT can be viewed as the implementation of powerful set of tools and system that process inputted digital information (from a variety of source) to reduce the uncertainty of some events by displaying or serving to determine a calculated outcome intended to better support an organization's plan of action.

All organizations implement IT in three ways:

1. Acquire and analyze information.
2. Act according to the interpretation of information.
3. Play an important role in managerial decision.

4. TRENDS IN INFORMATION TECHNOLOGY

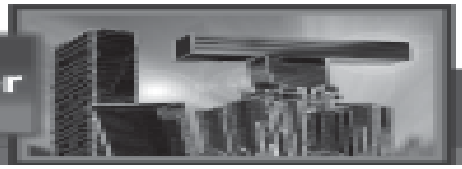
1. Corporate strategy is becoming dependent on Information Processing Technology.
2. It is an imperative part of the work place:
   a. reduce labour cost by automation.
   b. departmental consolidation of work effort on a computer network.
   c. Control production by software monitoring
   d. Increase volume of production in less time
   e. 24 hour remote access of organization files
   f. 24 hour open–access storefront through the www.
   g. Opportunity for telecommuting (Working at home or elsewhere)

3. Organization IT transforms

a. Individual Computer Workstations that enhance each employee production.

b. Cohesiveness in top to bottom effort via direct communication.

c. Re-engineering practices become easier to



implement

d. Cycle-times of data entry can be reduced by touch screen entries, voice recognition modules, optical character recognition (OCR) Scanning and bar-coding devices.

4. Technology creates Flexibility in an organization in the followings ways:

a. access to on-line services like Wall Street, quotes, libraries and Government database.

b. allow for remote login and telecommuting (Work can be done anywhere)

c. provide on-line customer services and technical support.

5. RESTRUCTURE OF THE IT CURRICULUM

Computer technology has been and is already playing a key role in re-engineering companies. Their trust is committed to delivering maximum efficiency, increase-production, lower cost and to promote high returns on equity. It is worth overstating that computer technology is being used by organizations to remain competitive in their respective fields. Independent studies have pointed out, that qualified and well-trained employees equipped with custom suited computer technology knowledge are being sought for to boast production morale and speed of task completion.

5.1 The Need For IT Orientation

Nigeria being a country that is just embracing IT, there is therefore a need to produce qualified computer technologists. In order to enable the Government IT policies of 2002 survive there is a need to inject IT knowledge into the blood of individual irrespective of their professional background. Today in the advance countries most transactions with Government, Organizations are being done through the computer (either by LAN or INTERNET). In America for example most transactions are done electronically-e-commerce, e-banking, e-business, e-government, e-learning etc. With the rate at which the Country is moving towards IT, in years to come non-IT compliant people will find it difficult to carry out some transactions. It must be noted that children, are not excluded from this facts.

6. ORIGIN FOR IT ORIENTATION

Most of the courses that are being offered in higher institutions of learning have their origins from the elementary level (Primary Schools/ Secondary Schools). In the present day Nigeria, computer science is majorly an higher institution's affairs with mathematical science origin from elementary level. For instance chemistry, physics are major subjects for medicine even has their main origin from Secondary School level.

The needed orientation for IT should commence from Secondary School level preferably Senior Secondary School level. This will serve as a guide to those who wish to take up career in IT. With this it is therefore imperative for the government to include or introduce IT into Secondary School as subject, which can then be taken in final examination like (WAEC-SSCE or NECO-SSCE).

6.1 Need for Uniform Curriculum

Curriculum in itself is a outline of courses to be taken or plan of work. It serves as a guide to students, lecturers and the entire education community. There should be a uniform curriculum for Secondary Schools and Higher Institutions of Learning. This will make it possible for students from different schools or background to undergo the same training and examination. Uniform Curriculum for IT will go a long way in making or sharing the same IT Ideas/Knowledge among students of IT from different schools.

7. COMPOSITION OF THE CURRICULUM

The IT curriculum under consideration should be structured as follows: The curriculum should cut across every area of IT. The IT is made up the followings: (1). Computer Knowledge (2). Information Processing and (3) Other electronics e.g. (Fax, ATM: Automatic Teller Machine).

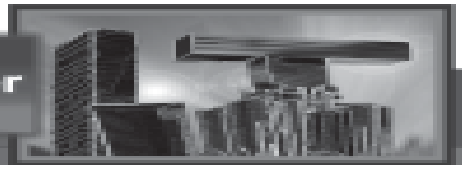
Computer Knowledge

The computer knowledge should be made up of (1). Computer Science Curriculum, which already is available in some higher institutions and as shown in table 1 above.

(2) the new aspects of IT such as Networking, Internet and Web Development (Net Technologies) (3) Software Engineering which should include both applications software and system software development. Network programming (Socket programming) is gaining popularity because of the increasing trend of computer network.

Software Engineering

This can be broken down into the followings:



Application Software Engineering, System Software Engineering, Socket programming e.g. Protocol development and System Analysis and Design.

Web and Internet Development

Because of the increase in the use of Internet and web by individuals, organizations and Governments, there is a need for those intending to professionalise in IT to be trained in these areas. Hence the following should be included in the exiting curriculum if any, otherwise proposed for the incoming curriculum which is yet to be design for our tertiary institutions in order to equipped our IT professionals with the followings knowledge in the areas enumerated below:

Server and Clients Technology, Web authoring tools e.g. (HTML, Scripting Languages), Hosting Technology, Web Maintenance and Internet Technology and Connectivity etc (Deitel

and Deitel, 2003).

Another area of importance that needed to be considered for this restructuring is the Networking of Computers: This is essential to IT world because IT was made possible through computer IT. The IT curriculum should be structure in a way that it will cut across every area of network: In the network aspects the followings are required:

Network fundamentals, Network Technologies, Setting up of LAN, (WLAN, WAN and INTRANET), Network Administration, Network Troubleshooting (repairs and maintenance), and Network Security.

Above all, most of the curriculum of other countries studied in the course of this work actually featured most of these important areas in their IT curriculums and nothing stop us in this country from embracing this ideas since it help in promoting the countries like Eastern Carolina, Canada etc., **In view of these, IT curriculum in Nigeria Tertiary Institutions can be restructure and improved on using the above trends in technologies** (Deitel and Deitel, 2002).

8. BENEFIT TO BE DERIVED

If the above structure can be adopted for IT curriculum, the country at large will have a lot to gain from it, if faithfully implemented, the followings stand out to be the benefits:

1. Increase in Manpower: Instead of using expatriates in some IT functions, there will be people available to perform such task in Nigeria.

2. Reduction in IT investment Cost: Many organizations have spent huge amounts in (1) Procurement of foreign software (2) Use of expatriates for some IT functions. Indigenous software that can compete with foreign ones can be produced or developed by trained IT professionals or graduate of IT discipline.

3. Unified Standard in all Institutions: This will make all institutions have the same curriculum for the IT training

4. It provide foundational Knowledge: If IT can commence and implemented in Secondary School, those who offered IT subjects in Secondary School level will form the basis for continuation in IT profession in higher institution level. This will definitely prevent those with non IT background to delve into IT area as a discipline or degree – result IT professions with full IT orientation there by making student to perform or meet up with challenges faced in industry after graduation.

9. CONCLUSION

From our survey and discussions so far we have been able to highlight most of the benefit that the nations and our institutions in general we derived from the IT curriculum restructuring, if only it is embraced and properly utilized. So far we have presented the current curriculum in use for the award of B.Sc.Computer Science in conventional Universities. Also, B.Sc. (MIS), Management Information System and B.Sc.Computer Science in one of the private University which implies the likely curriculum in other private Universities.

However, as far as the authors are concern, we have not heard or see currently any university here in Nigeria that runs B.Sc, Information Technology talk less of an already designed curriculum for IT. But, in view of our findings it is therefore imperative for Governments, Nigerian Computer Society (NCS), Computer Professional of Nigeria (CPN), ITAN, National University Commission (NUC) and other notable bodies to explore this area and see a crucial issue of this kind as featured in this conference can be actualized in our institutions as it were in other countries such as United State of American, Canada, Eastern Carolina, South Africa etc. The dream of restructuring should be made realizable in the nation at large, as it will benefits all and sundry.

10. RECOMMENDATION

Ones this dream is accomplished it is therefore recommend that the review of the IT curriculum should always be done at every two years interval to be able to meet up with the latest technological changes, since the field itself is dynamic. Also, the Government should always have interest in the derivative or outcome of

the skilled labour trained in such institutions. And see how they can be utilized in promoting the technological growth of the country. Finally, Tools and equipped laboratory should be provided for institutions that run such programme to enable them have comprehensive experience.

REFERENCES

Brandy P.D. et'al, (1999). Designing an Information Technology Curriculum, workshop on the emerging digital economy, Washington D.C. April 1999.

Deitel, H.M. et'al (2002), Internet and World Wide Web: How to Program, 2nd Edition , Prentice-Hall Inc., One lake Street, Upper Saddle River, New Jersey U.S.A.

Deitel, H.M. et'al (2003), Java: How to Program, Fifth Edition , Pearson Education Inc. One lake Street, Upper Saddle River, New Jersey,U.S.A.

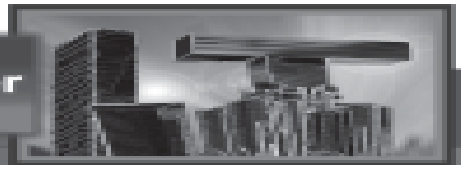
Falaki, S.O. (2002), "Information Technology in Nigeria Now or Never" Inaugural Lecture Series 29 delivered at Federal University of Technology, Akure, Ondo State. February 26th, 2002. pp. 16-18.

Fnsaau, (2004), Faculty of Natural Sciences, Ambrose Alli University, Ekpoma.

Students Hand book, by Aniko publisher, Benin city, Edo State.

Hardin, L. (1998), Information and Technology Curriculum: Similarities and Differences.

Oxford, D. (1999), The Oxford English Minidictionary, Fifth Edition, by Lucinda Coventry with Martin Nixon.



The Pervasive IT World: Fashioning out Enduring IT Curriculum - A Baseline Study for Capacity Building

Mike Ezeji

ABSTRACT

If you are defining computing, computers, computer science, data processing, information technology today, obviously your definitions must be different if you were defining it just few 15 years back. Many things have changed within the periods too rapidly and fast. Many medical, legal, engineering and environmental technologies and application have changed drastically, but principles did not as such. Example is that human anatomy and physiology from creation has been same but the great advancement in medicine has sought for better understanding and delivery of medical services. So is basic principles of fairness and equity, and laws of motion in law and science respectively have been same from creations but differs in delivery or application. This is not so in the pervasive Information Technology world which have lived too short a time compare to these professions.

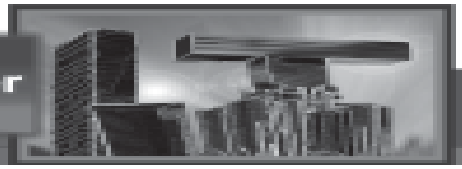
Computer, as a machine that **processes** data (input) to produce (output) information would have scored 100% correct in the 80's but today is no longer correct. It is now half-truth which is misleading as a lie. The scope, curriculum and practice of computer discipline have become extremely divergent. Fashioning out a long-term enduring curriculum for this kind of discipline is **not enviable** task but a **must** privilege for nation's building.

Presently, revisable Information Technology curriculums in Nigeria for years have been in search of passenger's seat. But this must move to become the substructure, bedrock of the NEXT economic capacity building. Information Technology cum Information Communication Technology has joined the league of most diversified professions. What used to exist before as computer made up of software and hardware is misleading and perhaps a sheer or mere academic exercise! IT paraded today (what used to be under two umbrellas of software scientist and hardware (electronic) engineer) a colossal and multifarious groups namely software development, system programming, systems maintenance/repair, system/network engineering, system/business support analysis, help desk, systems administrations/management, enterprise resource planning, database administration, management information system, Internet/intranet and applications, web designing/publishing, multimedia technologies, etc.

No one curriculum development can handle these and Information Technology academia only cannot be trusted with these responsibilities alone. This research suggests two-pronged revisable extensive systems to ensure fashioning enduring IT curriculum and collaborative systems to ensure fashioning enduring IT curriculum for making of IT capacity building economy. Revisable, because it will be crime to be static or expect a curriculum to last five years and collaborative because the academia cannot achieve it alone while the industry's primary business is not manpower development as it is academia.

Background of Study: Curriculum Differences in Different Societies

Curriculum, or what some societies narrowly called programme of study is a systematically organized course of teaching and learning (Mclean, 2004). Some definitions of "curriculum" focus narrowly on the arrangement of subjects over a sequence of grades; others include everything that students[1] and teachers[2] do.



Proper Information Technology Education will look at it from the academia and the professionalism. Curricula in different countries may be set and controlled centrally and standardized for each institution. They may vary between regions, localities, and institutions. There is clear and widespread agreement among the public and educators that all students and job seekers need to be proficient computer users or "computer literate." However, while some are spending a great deal of money on computer technology, there seems to be only a vague notion of what computer literacy really means.

Can the student who operates a computer well enough to play a game, chat, send e-mail or surf the Web be considered computer literate? Will a student who uses computers in school only for running tutorials or an integrated learning system have the skills necessary to survive in our IT society? Will the ability to do basic word processing be sufficient for students entering the workplace or post-secondary education?

Clearly not! In too many schools, teachers and students still use computers only as the equivalent of expensive flash cards, electronic worksheets, or as little more than a typewriter. The productivity side of computer use in the general content area curriculum is neglected or grossly underdeveloped (Moursund, 1995). In our country Nigeria, conservative 85% of computers are used for word processing and chatting (Ezeji, 1999).

Educational technologists are clearly describing what students should know and be able to do with technology. They are advocating integrating computer skills into the content areas, proclaiming that at best computer skills should not be taught in isolation and that separate "computer classes" do not really help students learn to apply computer skills in meaningful ways. There is increasing notion that the end result of computer literacy is not knowing how to operate computers, but to use technology as a tool for organization, communication, research, and problem solving. [If you listen well you will see some expressing such notion in this conference.] This is an important shift in approach and emphasis.

Moving from teaching isolated technology skills (as we have in various computer centres) to an integrated approach (as we will recommend) is an important step that takes a great deal of planning and effort for evolving enduring curriculum in pervasive IT world. Fortunately, this writer will propose a model for this. Researchers have shown information skills can be integrated effectively when the skills

(1) directly relate to the content area curriculum and to classroom assignments, and

(2) are tied together in a logical and systematic information process model.

Schools seeking to move from isolated information technology skills instruction will also need to focus on both of these requirements. Successful integrated information technology skills programs are designed around collaborative projects jointly planned and taught by academia[3] and professionals[4]. Information technology skills instruction can and should be imbedded in such a curriculum. Information Technology specialists and computer teachers need to work together to develop syllabus that will include technology skills, information skills, and content-area curriculum outcomes.

A meaningful, unified information technology literacy curriculum must be more than a "laundry list" of isolated skills, such as knowing the parts of the computer, writing drafts and final products with a word processor, and searching for information using the World Wide Web.

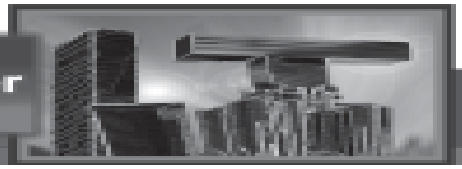
While these specific skills are important for students to learn, the "laundry list" approach does not provide an adequate model for students to research, to transfer and apply skills from situation to situation. This is the bane of computer centres. Their curricula address the "how" of computer use, but rarely the "when" or "why." Students may learn isolated skills and tools, but they would still lack an understanding of how those various skills fit together to solve problems and complete tasks. Students need to be able to use computers and other technologies flexibly, creatively and purposefully. All learners should be able to recognize what they need to accomplish, determine whether a computer will help them to do so, and then be able to use the computer as part of the process of accomplishing their task. Individual computer skills take on a new meaning when they are integrated within this type of information problem-solving process, and students develop true "information technology literacy" because they have genuinely applied various information technology skills as part of the learning process. Some technology literacy competencies that may be relevant in some situations include:

(1) knowing the basic operation, terminology, and maintenance of equipment,

(2) knowing how to use computer-assisted instructional programs,

(3) having knowledge of the impact of technology on careers, society, and culture (as a direct instructional objective), and

(4) computer programming.



The Pervasive Task for Pervasive IT: Integration of Cross-Curricular Interests

Throughout the curriculum development and revision process, the advice of experts from both academia and professional should be sought after for cross-curricular integration to ensure relevance in this fast lane discipline called IT. The prescribed goals for Information Technology can be grouped into the following four:

- Foundations/Appreciation
  - Basics, Introduction or Computer Appreciation
  - Maintenance, Repairs of Hardware and Software
  - Network Planning
- Programming
  - Programming Languages
  - Object Oriented Programming
  - Structured Programming
- Process
  - Network Setup
  - Network Building
  - Electronic Communications
- Presentation
  - Multimedia

A. Foundations

Foundations provide students with the fundamental knowledge, skills, and attitudes needed for a lifetime of using information technology. Issues of ergonomics, ethics, and the safe use of tools are included, as are connections to larger social issues such as security of information, copyright, and personal freedom. The prescribed learning goals should emphasize:

- acquiring appreciation skills for using information technology tools
- developing the knowledge and skills to formulate questions and to access and retrieve information from a variety of sources
- exploring careers and occupations related to information technology
- developing suitable attitudes and practices about safety and ergonomics in the use of information technology tools
- developing an understanding of the ethical use of information technology tools
- developing a positive attitude toward using information technology as a tool for lifelong learning
- integrating and applying these skills across all areas of learning

i. Computer Appreciation

Computer Appreciation focuses on Computer fundamentals, history, trends in computing, computer parts, and devices e.g. printers, CDROMs, software and hardware introductions, etc

ii. Maintenance and Repair

These are designed to impart Software and hardware installation, setup, maintenance and repairs, brands and cloning (assembly) of systems

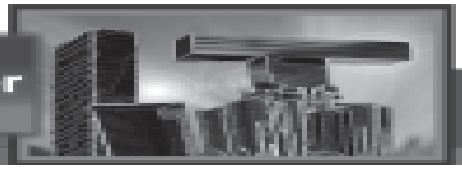
iii. Network Planning

This is designed with the challenge to construct and manage networks that contribute to the quality of life within an ethical framework, how it affects the free-flow of ideas as well as the freedom and privacy of information. The above prescribed learning outcomes emphasize:

- using, planning, and creating networks
- the concept of retrieving digital information using small networked computer systems
- ethical considerations regarding privacy and internalizing standards for appropriate use (awareness or abuse)
- manipulating the components of LAN to create the best solution for a specific need
- setting up networks

B. Programming

Programming is the process of creating logical, executable steps for a computer to perform a desired task. Students must learn computer language and logical thinking, and use a problem-solving model that includes the process of top-down design. The prescribed



learning goals emphasize:

- writing fluent and logical programs using higher-level language
- acquiring the skills and knowledge to solve problems using a structured approach and algorithms

C. Process

Process allows students to select (query, retrieve), organize, and modify information to solve problems. Students develop skills in selecting appropriate information technology tools, and they learn to use these tools to access and structure information to analyse problems, synthesize ideas, and justify opinions or values. Students also gain an understanding of time, resource, and project management. The study goals emphasize:

- evaluating and selecting information based on specific criterion
- developing information literacy by accessing, evaluating, synthesizing, making inferences, validating, and creating information using appropriate information technology tools
- statistics
- database management system
- understanding the ethical use of information

i. Enterprise Resource Planning [ERP]

This focuses on building networks as first member of ERP, network software, enterprise resource planning software setups and installations, electronic messaging, real life network, using different protocols, topology, types, etc.

ii. Electronic Communications

With Electronic Communications, students develop an awareness of the globalization of information and ideas, and gain mastery of electronic communications concepts and skills. This is learned as they explore the Internet and develop expertise in the ethical use of electronic communications tools. The prescribed study goals emphasize:

- understanding the social and ethical principles associated with electronic communications
- E-services – www, e-mail, ftp, gopher, telnet, usenet, etc.
- acquiring the skills and knowledge to select information technology resources from a wide variety of information sources
- developing the skills to produce and present materials for electronic distribution
- using electronic communications to solve problems in students' daily lives

D. Presentation

Presentation provides students with an understanding of how to communicate ideas effectively using a variety of information media. In addition to learning the principles of effective communication, students develop skills in integrating text, graphics, and audio to communicate to a specific audience. The prescribed study goals emphasize:

- developing an understanding of digitized media
- applying the principles of communication and design to develop an effective presentation
- using a variety of information technology tools to synthesize the presentation of ideas and information eg demos, slides
- thinking critically to determine and develop the most effective media for presenting ideas and information to an audience

i. Multimedia

Multimedia allows students to exchange

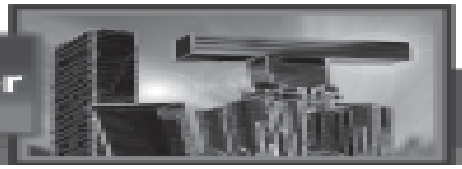
References

Britanica Encyclopedia 2004

Ezeji M O, Management and Regulation of Information Communication Technology: Issues of Ethics, Liabilities and Computer Laws published on COAN INFOCOMM '99 conference proceedings Page 227 to 231

Microsoft Encata Premium Suite 2004

www.bced.gov.bc



ADAPTING IT-CURRICULUM TO CHANGE

Okoronkwo Matthew Chukwuemeka, Prof. Inyiama Hyacinth. C.

ABSTRACT

Change is as old as creation itself, and it is one basic law, which every imperfection must obey and adapt to in the quest for perfection. Man as part of creation have not excluded himself from change and so much so his own developments/inventions. Rather constant efforts are put into adapting himself and his endeavours to change in an effort to remain relevant; otherwise, he quickly becomes part of history. The professionals in this field are seriously being challenged to harness the potent force of ICT to develop/create opportunities, which would eradicate poverty through empowering IT-graduates. They must recognise that our IT-graduates are the future workforce, leading creators and adopters of ICTs. They must therefore be empowered as learners, developers, contributors, entrepreneurs and decision-makers to take the centre stage. This paper takes a hard look at the impact of IT the world over, the characteristics of our IT-graduates; how they are coping with the rate of change in ICT and how our IT-curriculum has contributed to these characteristics. It also suggests ways of restructuring the curriculum to be that tool that will be used to produce quality professional who will be constantly in tune with the changes in IT-technology and contribute in determining/setting the pace. It looks at the IT-curriculum as a benchmark that should be restructured by standing committees drawn not just from educational institutions and worst off from the aging generation in the profession who hardly find time to look around to observe events but from various arms of IT-Community. These committees should include distinguished academics current in their fields and are IT-compliant, notable figures in IT-Industry, IT-Product vendors and enlightened members of the general public especially major employers of labour. Each of the various committees should be able to meet quarterly to review changes in IT and recommend ways of updating the IT-Curriculum. Subsequently, the recommendations of the various committees are collated and harmonised leading to emergence of a fresh IT-Curriculum.

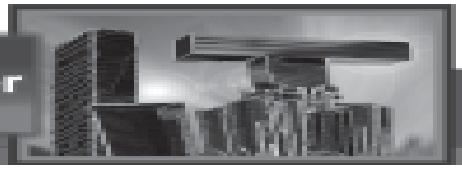
1.0 INTRODUCTION

In everything and for everyone the law of growth is through change. The ability of an individual to adapt himself to all the circumstances, which affects his life, is one major factor of an attractive personality. It is also the cranium of adaptability to the great law of growth through change.

Information and communication technology ICT impels change. The rate of ICT adoption across faculties and disciplines is likely to be on the increase for quite some time in the future, thus increasing the need for investments in areas such as new equipment, more training, new courses and other challenges. Infrastructure can drive the innovation which in turn creates still more demand for Infrastructure and support as growing number of disciplines adopt and adapt ICT resources in their activities. As Ramsden (1998) states "it is not so much the technology that are critical, but the underlying concepts of versatility in time and place of learning, and new ways of thinking about the human aspects of teaching and learning, which these techniques can make more easily reliable".

ICT places new demands on both students and teachers such as learning of new skills in developing and maintaining course and assessment materials and spending time differently. As Fox (2000) asserts "new technologies will not allow us to easily or more

effectively do the same things the way we are used to, rather it will change what we do, our work practices and relations, our jobs and our futures. It will also change what and how students learn".



The challenge is how to regulate and adapt what and how students learn to the changing technology and this is the issues discussed in this paper. For our educational institutions the regulations are spelt out in curriculum benchmarks, the extent to which we are able to make the IT curriculum flexible with change determines the extent to which ICT permeates institutions and various endeavours in this country.

Evidence abound that ICT applications have not penetrated our society more than superficial level, and the level of expertise and practice is not yet sufficient to ensure wider application. So the potent force of ICT to develop/create opportunities, which would eradicate poverty through empowering IT-graduates, will continue to elude this country and place her at the wrong side of the digital divide. Computer science curriculum and education must seek ways to prepare students for life long learning that will enable them to move beyond today's technology to meet the challenges of the future.

1.1 The Background

It is well known or generally accepted that the World is in the Information age, an era characterised by the electronic transmission of information. The technologies have brought about educational concepts such as "flexible modes of delivering lectures', 'open learning', 'life long learning', 'virtual classrooms' and institutions without walls'. Newer technologies involving audio and video tapes, computers, computer-based learning packages, interactive video and multimedia, audio graphic communication systems, and video conferencing have now surfaced and are being applied in most institutions in developed countries for teaching and learning.

In this part of the world how do we stand? Even the teacher's adoption level of ICT is still far from appreciation not to talk of application. Since one cannot give what he does not have, the products therefore cannot be any better.

It could be recalled that in past conferences of NCS, students have questioned the relevance of some courses offered to computer science students and how this clarion call for a change have been shoved aside. This paper thinks that it is high time this issue is properly discussed and corrective action taken to address it. We cannot continue to be satisfied only with knowing our past and forgetting the present, which holds more promises for us as a group and the country to reap the benefits of mass adoption of the computing technologies.

Those of us here who are not from educational institutions know the products of our tertiary institutions better than the teachers. I am sure that if you ask them, they would sincerely tell you that the performances of these graduates are far from satisfactory, and they are forced to invest much in retraining. The question is why can't our institutions save them the extra cost and the graduates the embarrassments they face at interviews. It is high time they were integrated in the review of our curricula for computer science programs.

2.0 Innovation Uptake

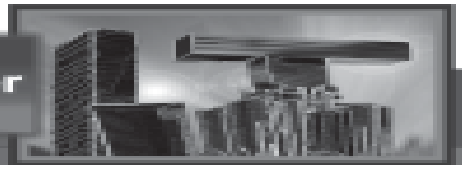
A well-known model concerning the diffusion of educational innovations has been based on the work of Rogers (1995). He identified categories of innovation uptake from high level through to low-level innovators, early adopters, early majority, late majority and laggards. Under this model he asserts that for a significant change to occur, a "critical mass" of individuals need to have adopted and implemented a given innovation (Green & Gilbert 1995). This "Critical mass" occurs when enough individuals have adopted the innovation so that the innovation further rate of adoption becomes self-sustaining. Unfortunately, we are yet to get there or should I say rather that the enabling investments have not been sufficiently made at the right places. In addition to the critical mass factors, there are pedagogical forces that have driven the push for educational institutions to adopt and incorporate information technologies. These include access to information, open learning and communication skill.

Access to Information

The World Wide Web has made it possible for all people to have access to primary sources of information. This tool if mastered, becomes essential in gaining access to ever growing body of recent and up to date knowledge available to everyone, electronically. This body of knowledge is there unexplored. How many teachers including computer teachers have found the time to understand the tool and benefit from its immense resources?

Open learning

Institutions are now enabled to cater for varieties of students by removing the barriers of time and distance. Students who would have been disadvantaged by geographical barrier now have access to variety of resources not usually at their disposal (Deden & Cater, 1996). How many of our teachers and students benefit



from this, except perhaps those in the Open University system?

Expected Communication skills

Employers of labour expect graduates to be familiar with the basic applications of computers, email etiquette and associated communications tools. The lack of some of these most basic skills has forced some employers to demand foreign certifications as basis for invitation of prospective employees for interviews. This implies that they are losing confidence in our graduates and it is a dangerous trend we must check.

2.1 The Inhibiting Factors

Even with the obvious benefits of ICT, the question which any concerned person would not fail to ask is why haven't these new and powerful technologies permeated our institutions a greater extent? A variety of factors highlighted below contribute to this lack of adoption and effective deployment of ICT at tertiary institutions.

Leadership

According to Dolence & Norris (1996), many educational leaders are inexperienced in conceiving growth oriented learning opportunities in the Information Age. The lack of models for integrating ICT into the curriculum (Schofield, 1995; Gilbert, 1996; Northrup, 1997) also contributes to a lack of effective institutional planning. They have been forced to deal with constant patterns of reorganisation and restructuring involving the difficult task of redefining values and transforming the culture of their organisations (Middlehurst, 1995; Farmer, 1990).

Other institutional problems says Gilbert (1996) point to fragmented institutional planning where institutions fail to match the technology investment with an investment in people (i.e. adequate training, appropriate incentives). In other institutions, plans tend to be driven by information technology and not necessarily by a pedagogical rationale and focus (Deden & Carter, 1996; Gilbert, 1996; Brown, Burg & Dominick, 1998).

Technology infrastructure and cost:

Lack of uniformity in computer hardware and software systems (Brown, Burg & Dominick, 1998) within the one organisation is another factor noted in the literature as hindering the adoption of ICT. There also appears to be an assumption that technology will reduce operating costs and increase productivity (Green & Gilbert, 1995).

Attitude to change:

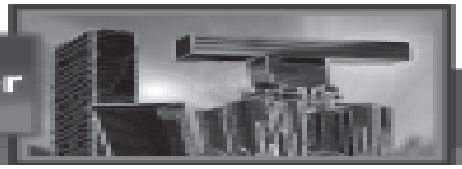
It should be noted that one of the major factors contributing to the lack of adoption of any innovation, not just information technology, is the entrenched attitudes of the teaching staff along with an associated dysfunctional behaviour (i.e. resisting change).

Resources:

One of the major concerns for most institutions is lack of resources and funding. People resources are stretched to the limit and teaching staff are not only asked to do more, but they are expected to do it differently (Gilbert, 1996; Northrup & Little, 1996).

Universities cannot promote the use of ICT in teaching and learning without adequate support through funding. And as more people adopt ICT the support needs are simply going to increase. It would seem that a project team model such as the one advocated by Bates (2000) would assist with many of these support issues. The unit could comprise of one technical support person (1:25 staff utilising ICT for teaching) and one generalist educationalist technologist (1:50 academic staff) as strongly suggested by Bates (2000). The challenge, which we need to address, is how to prepare these support persons to be able to face the challenges of rapid change in IT. The starting point is the curriculum, which stipulates what should be given especially, to computer science graduates.

If ICT is to be seen as an integral part of the teaching and learning process then the University must provide ample computer access and projection facilities in all teaching classrooms. Remote access facilities for staff need to be provided free of charge and Departments/Schools should be encouraged to support leasing laptops for staff instead of stand alone computers. More importantly, there must be conscious effort to update the teachers. These will enable staff to take advantage of the full flexibility of ICT.



3.0 The Challenging Impact of IT

It will be stating the obvious to say that any profession/ industry that ignores IT is simply heading towards extinction. ICT has turned the World into a global village, breaking all barriers and offering strong information base to all disciplines and human endeavours, and the following are some of the consequences.

- National corporations are becoming international.
- The culture we live in is becoming cosmopolitan, that is, belonging to the whole world.
- The issues confronting one nation are increasingly international.
- Issues of economic competition, the environment, and the movement of peoples around the world require an awareness of political associations that are larger in scope than the nation-state.

3.1 The consequences

i. **Greater demand for high quality labour forces,** which is versatile and adapts quickly to changing industrial requirements. To meet this demand the education and skill delivery system will be made more responsive to the requirements of a dynamic labour market. Education should emphasise the inculcation of positive and progressive values, including good work ethics and industrial discipline.

ii. **Labour force with broad-based education** emphasizing communications technologies. This attribute will provide the foundation for trainable labour force, which can adapt swiftly to the changing technological needs of the country. In this regard, improvements in IT curriculum should be a continuous process and more investments should be made in expanding facilities and enhancing quality of teachers in tertiary institutions.

iii. **There has to be regular and close monitoring of the curricula** of the educational and training institutions to meet the demand for new skills arising from rapid technological developments. Educational and training facilities must adapt quickly to facilitate efforts in producing suitably qualified manpower. The efforts of governments in this regard, should be complimented by private sector participation both in the areas of curriculum development and in the provision of training facilities.

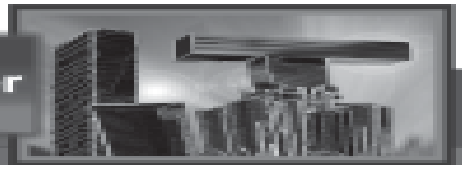
iv. **Greater demand for critical industrial skills** such as tooling and fabrication of precision parts and components, which are becoming more automated and complex. Training institutions must be equipped to produce such types of manpower, particularly those related to electronic and computer-based skills. In developing these skills, emphasis should be placed on leadership qualities such as creativity, innovativeness and entrepreneurship.

4.0 The ideal characteristics of computer science graduates

The issues which determine whether a specific program meets the necessary requirements to be called a computer science degree program are concerned with the overall nature of the discipline, the breadth and depth of a program, plus other factors relating to practical, personal, and transferable skills.

In general terms, institutions are expected to define outcomes and objectives that characterize their particular programs and ensure that their curricula are at the level of an undergraduate degree in computer science. Degree programs in computer science can take various forms, each of which could prepare students for different but valid careers. At one extreme, a degree program might provide opportunities for students to take courses on a wide range of topics spanning the entire area of computer science. Graduates from such programs would have great flexibility and might be of particular value either in emerging areas where specialist courses may not be established or in contexts where their ability to span the field would be useful. At another extreme, a program might take one very specific aspect of computer science and cover it in great depth. The graduates from such programs would typically tend to seek opportunities in the area of specialization they have studied, whether it be the development of multimedia systems, network design, formal verification for safety-critical systems, electronic commerce, or other specialties that emerge and become important. Despite such differences in emphasis and content, however, there are certain minimal characteristics that are expected of any graduate in computer science. The purpose of this chapter is to explore what those characteristics should be.

The material here draws heavily on a report designed to identify the desired characteristics of computer science graduates in the United Kingdom (see www.sigcse.org/cc2001/cs-references.htm#QAA2000). The objective is to define standard thresholds that all graduates of computer science programs are expected



to achieve. It says that Looking at the objectives of an academic program in terms of the characteristics of its graduates makes it easier to design assessment measures that ensure that those objectives are being met.

While the characteristics that one expects of graduates are related to the learning objectives associated with the core units, the expectations one assigns to those who complete an undergraduate degree in computer science reflect a more global level of student achievement. The learning objectives should specify what a student must know at the conclusion of any particular unit. The goal is to identify the characteristics that a successful graduate should possess. At a broad level, these characteristics can be expressed as follows:

- *System-level perception:* Graduates of a computer science program must develop a high-level understanding of systems as a whole. This understanding must go beyond the implementation details of the various components to include an appreciation for the structure of computer systems and the processes involved in their construction and analysis.
- *Appreciation of the relationship between theory and practice:* A fundamental aspect of computer science is the balance between theory and practice and the essential link between them. Graduates of a computer science program must understand not only the theoretical aspects of the discipline but also how the theory influences practice.
- *Familiarity with common themes:* In the course of an undergraduate program in computer science, students will encounter many recurring themes such as abstraction, complexity, and evolutionary change. Graduates should recognize that these themes have broad application to the field of computer science and must not compartmentalize them as relevant only to the domains in which they are introduced.
- *Significant project experience:* To ensure that graduates can successfully apply the knowledge they have gained, all students in computer science programs must be involved in at least one substantial software project. Such a project demonstrates the practical application of principles learned in different courses and forces students to integrate material learned at different stages of the curriculum.
- *Adaptability:* The essential characteristics of computer science have been an enormous pace of change. Graduates of a computer science program must possess a solid foundation that allows them to maintain their skills as the field evolves.

4.1.0 Capabilities and skills

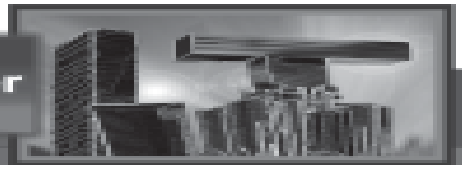
Students of computer science must develop a wide range of capabilities and skills. Some of these skills are specific to degrees in computer science; others are more general and would be expected of any graduate of a technical discipline. These capabilities and skills may be divided into three general categories;

- Cognitive capabilities relating to intellectual tasks specific to computer science
- Practical skills relating to computer science
- Transferable skills that may be developed in the context of computer science but which are of a general nature and applicable in many other contexts as well

The required capabilities and skills are outlined below. In each case, the institution must ensure that the skills in each of these categories - cognitive, practical, and general receive sufficient coverage and that all students have had the necessary background prior to graduation.

4.1.1 Cognitive capabilities and skills relating to computer science

- *Knowledge and understandin:.* Demonstrate knowledge and understanding of essential facts, concepts, principles, and theories relating to computer science and software applications.
- *Modelling:* Use such knowledge and understanding in the modelling and design of computer-based systems in a way that demonstrates comprehension of the trade-off involved in design choices.
- *Requirement:.* Identify and analyse criteria and specifications appropriate to specific problems, and plan strategies for their solution.
- *Critical evaluation and testing:* Analyse the extent to which a computer-based system meets the criteria defined for its current use and future development.
- *Methods and tools:* Deploy appropriate theory, practices, and tools for the specification,



design, implementation, and evaluation of computer-based systems.

- *Professional responsibility:* Recognize and be guided by the social, professional, and ethical issues involved in the use of computer technology.

4.1.2 Practical capabilities and skills relating to computer science

- *Design and implementation:* Specify, design, and implement computer-based systems.
- *Evaluation:* Evaluate systems in terms of general quality attributes and possible tradeoffs presented within the given problem.
- *Information management:* Apply the principles of effective information management, information organization, and information-retrieval skills to information of various kinds, including text, images, sound, and video.
- *Human-computer interaction:* Apply the principles of human-computer interaction to the evaluation and construction of a wide range of materials including user interfaces, web pages, and multimedia systems.
- *Risk assessment:* Identify any risks or safety aspects that may be involved in the operation of computing equipment within a given context.
- *Tools:* Deploy effectively the tools used for the construction and documentation of software, with particular emphasis on understanding the whole process involved in using computers to solve practical problems.
- *Operation:* Operate computing equipment and software systems effectively.

4.1.3 Transferable skills

- *Communication:* Make succinct presentations to a range of audience about technical problems and their solutions.
- *Teamwork:* Be able to work effectively as a member of a development team.
- *Numeracy:* Understand and explain the quantitative dimensions of a problem.
- *Self-motivated:* Manage one's own learning and development, including time management and organizational skills
- *Professional development:* Keep abreast of current developments in the discipline to continue one's own professional development.

4.2 Coping with change

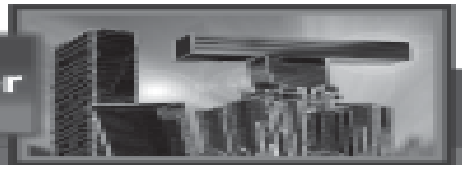
An essential requirement of any computer science degree is that it should enable graduates to cope with and even benefit from the rapid change that is a continuing feature of the computing field. But how does one achieve this goal in practice? At one level, the pace of change represents a challenge to academic staff that must continually update courses and equipment. At another level, however, it suggests a shift in pedagogy away from the transmission of specific material, which will quickly become dated, toward modes of instruction that encourage students to acquire knowledge and skills on their own.

Fundamentally, teaching students to cope with change requires instilling in them attitudes that promote continued study throughout a career. To this end, a computer science curriculum must strive to meet the following challenges:

- Adopt a teaching methodology that emphasizes learning as opposed to teaching, with students continually being challenged to think independently.
- Assign challenging and imaginative exercises that encourage student initiative.
- Present a sound framework with appropriate theory that ensures that the education is sustainable.
- Ensure that equipment and teaching materials remain up to date.
- Make students aware of information resources and appropriate strategies for staying current in the field.
- Encourage cooperative learning and the use of communication technologies to promote group interaction.
- Convince students of the need for continuing professional development to promote lifelong learning.

5.0 Benchmark standards

In seeking to define an appropriate set of objectives for computer science graduates, the authors of the UK benchmarking report (check www.sigcse.org/cc2001/cs-references.htm#QAA2000) recognized that



establishing a minimum standard may discourage both faculty and students from pushing for excellence beyond that minimum. To avoid this danger, the report provides benchmarking standards to assess various levels of achievement. At the lowest level, the report identifies a *threshold standard* consisting of a set of objectives that any graduate must be able to meet. The report goes on to identify a somewhat *modal standard* corresponding to the expected level of the average student.

Defining objectives for the threshold and modal standards represents a valuable opportunity for a department engaged in undergraduate computer science education. Setting such objectives makes it easier to understand the overall impact of the curriculum and makes it possible to assess the effectiveness of the educational process. While these objectives will vary by the type of program and the characteristics of individual institutions, the objectives shown below in the threshold standard, which are adapted from the UK benchmarking report, may provide a useful model for our local implementations.

5.1 Minimum standards

- Demonstrate a requisite understanding of the main body of knowledge and theories of computer science.
- Understand and apply essential concepts, principles, and practices in the context of well-defined scenarios, showing judgment in the selection and application of tools and techniques.
- Produce work involving problem identification, analysis, design, and development of a software system, along with appropriate documentation. The work must show some problem-solving and evaluation skills drawing on some supporting evidence and demonstrate a requisite understanding of and appreciation for quality.
- Demonstrate the ability to work as an individual under guidance and as a team member.
- Identify appropriate practices within a professional, legal, and ethical framework.
- Appreciate the need for continuing professional development.
- Discuss applications based upon the body of knowledge.

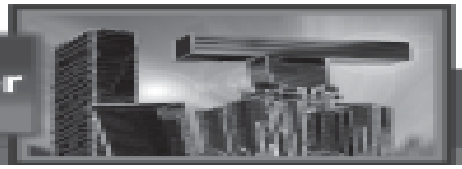
5.2 Average standards

- Demonstrate a sound understanding of the main areas of the body of knowledge and the theories of computer science, with an ability to exercise critical judgment across a range of issues.
- Critically analyse and apply a range of concepts, principles, and practices of the subject in the context of loosely specified problems, showing effective judgment in the selection and use of tools and techniques.
- Produce work involving problem identification, analysis, design, and development of a software system, along with appropriate documentation. The work must show a range of problem solving and evaluation skills, draw upon supporting evidence, and demonstrate a good understanding of the need for quality.
- Demonstrate the ability to work as an individual with minimum guidance and as either a leader or member of a team.
- Follow appropriate practices within a professional, legal, and ethical framework.
- Identify mechanisms for continuing professional development and life-long learning.
- Explain a wide range of applications based upon the body of knowledge.

Even though these benchmark standards are defined only for the minimum and the average, it is nevertheless important for programs in computer science to provide opportunities for students of the highest caliber to achieve their full potential. Such students will be creative and innovative in their application of the principles covered in the curriculum; they will be able to contribute significantly to the analysis, design, and development of systems, which are complex, and fit for purpose; and they should be able to exercise critical evaluation and review of both their own work and the work of others. Inasmuch as human ingenuity and creativity have fostered the rapid development of the discipline of computer science in the past, programs in computer science should not limit those who will lead the development of the discipline in the future.

6.0 The way forward

Curricula should adapt to changes in technologies within the shortest possible time. It should undergo continuous review by standing (not ad-hoc) committees with members drawn from the entire stakeholders in computer and Information technology. In selecting



members, care must be taken to choose people who are visibly compliant with the state of the art in these technologies.

While the Nigerian Universities/National Board for technical Education (NUC/NBTE) remain the pivot, they should draw terms of references for the various committees, define the standards of their operations and collate/harmonise the work of the various committees for the production of curricula for various tertiary institutions (not benchmark).

The committee should go beyond developing benchmarks to defining outcomes, objectives and the specific contents for a particular computing career. The task of developing the detailed program contents should no longer be the responsibility of individual school. Experience has shown that many institutions do not have what should pass for a curriculum for most of their program; some have good curriculum on paper but hardly operate it. The delivery of some curriculum is brought down to the level of the lecturer who decides what to teach. The decision as to what he teaches is again dependent on a number of factors that are at times selfish motivated or due to limitations imposed by lack of up dated knowledge

What this paper is advocating is the development of curriculum for each specialised areas of computing; Computer Science (CS), Information Technology (IT), Computer and Information Technology (CIT) and Computer science and Management Information Systems (CSMIS), and other emerging specialties etc. The curriculum should be handed down to institutions not to tinker with but to operate. This also implies that the era of a graduate know it all should give way to compartmentalised or specialised area, which a student should be prepared to undertake and grow along with. This paper thinks that it is enough with the present system that produce only half-baked graduates, with the consequential increases in retraining costs.

This issue has become more serious as new dimensions are continuously emerging in ICT, necessitating expansions in existing curriculum to accommodate them. So for how long can we continue to do this? The students are thereby encumbered with excess course loads without the enabling facilities required to effectively deliver. The graduates are worst for it. They are assumed to possess knowledge, which in reality they were not taught or at best the theoretical aspects delivered while the important practical components are left out. This is why today one is not surprised to see computer graduates that cannot operate computers, this should not be allowed to continue.

6.1 The Composition of the Curriculum standing Committees.

The suggested composition of the committees should comprise representatives of the following stakeholders:

1. Lecturers (academics),
2. NCS and other professional bodies in ICT,
3. Key players in major industries,
4. IT product vendors,
5. Nigeria Information and Technology development Agency (NITDA),
6. Nigerian Univ. Commission /National Board for Tech. Education (as coordinators),
7. Ministries of Science & Technology and that of Education,
8. National Association of Computer Science Students (NACOSS).

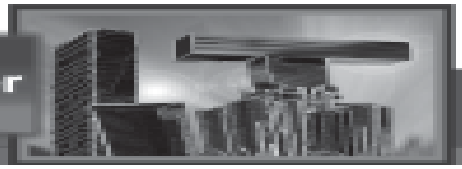
In constituting the various committees, conscious effort should be made to select representatives who are ICT compliant by direct involvement (not because of position or personality). One who plays key roles in more than one area as listed above should be preferred. For instance, one in the industry as an employer of labour and in IT-product vendor, plays two important roles. Such a person has more stakes and is better positioned to offer more meaningful contributions.

In carrying out their functions, they should request the contributions of interested groups through such mediums like questionnaires, Radio/TV programs, seminars/workshops where lecturers/students are especially encouraged to be active participants. Contributions that help to feel the pulse of the people should form part of the yardstick to measure impact and should be used as benchmark to further improve operative curricula.

7.0 Summary/Conclusion

In this paper, we have argued the fundamental importance of well-crafted Computing curricula, presented three distinct types of courses (majors) that departments may want to consider, presented ideal qualities of a computer science graduate and suggested ways to go about in reviewing curricula.

When designing and developing these courses, computer science faculty must be mindful of the needs of the intended audience and carefully align these courses to meet these needs. It must not do this in isolation, instead, it should seek out the contributions



of colleagues outside, when developing the goals, content, learning activities, and outcomes of these courses.

Given the pace of change in our discipline, the process of updating the curriculum once a decade has become unworkable. The professional associations in this discipline must establish ongoing review processes that allow individual components of curricula to be updated yearly.

Over the last decade, computer science has expanded to such an extent that it is no longer possible to add new topics without taking away some. So breaking computer science into majors and developing students along such majors is being advocated.

Computer science curriculum should seek to prepare students for lifelong learning that will enable them to move beyond today's technology to meet the challenges of the future.

Reference:

Baldwin, R. (1998). Technology's impact on faculty life and work. In K. Herr Gillespie (Ed), *New Directions for Teaching and Learning: The Impact of Technology on Faculty Development, Life, and Work.* 76, p7-22, Jossey-Bass: San Francisco.

Bates, A. (2000). *Managing technological change: Strategies for college and University leaders.* San Francisco: Jossey-Bass Publishers.

Candiotti, A., & Clarke, N. (1998). *Combining universal with faculty development and academic facilities*. Communications of the ACM*, 41*,(1), 58-63.

Curtin University of Technology, (2000) *Overview of the LEAP Project.* [On-line]. Available: http://otl.curtin.edu.au/leap/overview.html

Deden, A. (1998). *Computers and systemic change in higher education*. Communications of the ACM, 41,(1), 58-63.

Deden, A., & Carter, V. (1996). *Using technology to enhance students' skills*. In E. A Jones (Ed.), *Preparing competent college graduates: setting new and higher expectations for student learning* (pp 81-92). Jossey-Bass Publishers: San Francisco.

DETYA (2000). *Learning for the knowledge society: An education and training action plan for the information economy.* Department of Education, Training and Youth Affairs. Canberra: Australian Government Publishing Service.

Dolence, M., & Norris, D. (1995). *Transforming Higher Education: A Vision for Learning in the 21st Century.* Society for College and University Planning: USA.

Fox, R. (2000). *Online technologies changing universities practices*. In A. Herrmann and M. Kulski (Eds.*), Flexible Future in University Teaching. Centre for Educational Advancement,* (pp235 – 243). Curtin University of Technology: WA.

Gilbert, S. (1996). *Making the most of a slow revolution*. Change, 28,(2), 10-20.

Green, K. (1999). *Campus, computing survey: a national study of the use of information technology in higher education*. The Campus Computing Project: Encino, California. [On-line]. Available: http://www.campuscomputing.net

Green, K., & Gilbert, S. (1995). *Great expectation: Content, communications, productivity, and the role of information technology in higher education. Change, 27,(*2), 8-18.

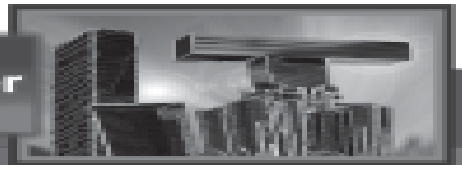
Jacobsen, D. (2000). Examining technology adoption patterns by faculty in higher education. In Proceedings of ACE 2000: Learning technologies, teaching and the future of schools. July 6 - 9, Melbourne, Australia. [On-line]. Available: http://www.ucalgary.ca/~dmjacobs/acec/index.html

Ramsden, P. (1998). *Learning to Lead in Higher Education.* London: Routledge.

Rogers, E. (1995). *Diffusion of Innovations,* 4th Edition. The Free Press: New York.

Schofield, J. (1995). *Computers and classroom culture.* Cambridge, NY: Cambridge University Press.

World summit on the Information society, Geneva 2003 WSIS-03/GENEVA/DOC/4-E. *Building the Information Society: a global challenge in the new Millennium.*



Using IT as A Subject, Tool for Curriculum Development in the National Open University of Nigeria

Lady Mercy Ogoegbunam Adibe

Abstract

The institutional and organizational structure of higher education is changing to emphasize academic accountability, competence outcomes, outsourcing, content standardizing and adaptation to learner consumer demands. For educators at all levels who are preparing students for information age, the challenges of introducing and integration IT into education have become even more challenging. The teacher of the future must not only be accomplished in the use of IT but also in the integration of IT into the curriculum. It is therefore inevitable that for any effective change to occur the use of IT in teaching and learning must be accompanied by a corresponding change in curriculum. The integration of information technology into school curricula is almost invariably a key element of initiatives in response to these challenges. The discussion above has highlighted the importance of integrating IT into the curriculum to enhance learning outcomes. Clearly, the curriculum must be adapted or redesigned so that it is ready for IT integration. Nigeria has taken a bold step in establishing the National Open University of Nigeria to represent the distant education at University level with laudable blue print on IT implementation yet undeniably IT standardization is at a critical crossroad. The question is not whether to fix the situation. It has to be fixed. The question is how to fix it. This paper examines the use of IT as a subject, tool for curriculum development.

INTRODUCTION

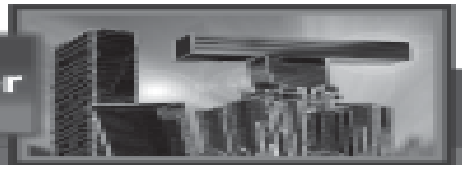
As sophisticated electronic technologies become standard equipment in schools, the roles of students and teachers will change. Before this change can take place it is critical that teachers accept the use of IT for classroom. Concurrently school management teams need to plan using IT for teaching, learning and management and provide as much scaffolding as possible to ensure that teachers interpret the prospects of technological innovations in a positive manner because schools have been known to be resistant to curriculum innovations.

Information and Communications Technology (ICT) offers new and innovative modes of learning for all students at all educational levels. ICT can bring about classrooms without walls when teachers are ready to realize the potential of this powerful tool.

However, at the present time, despite the technology changes in society, teachers in schools are still to a great extent using the approach of helping students acquire information from textbooks and acting as the information giver. All of their formal teaching in classrooms is still driven by traditional teaching practices although there may be occasions when ICT is used.

Breaking away from traditional approaches to instruction means taking risks and this is not easy for teachers to do on their own. It takes time for teachers to recognize the value of using ICT and to become fully committed to the view that information is available from sources that go well beyond textbooks and themselves. As change agents they must help students understand and make use of the many ways in which they can gain access to information and how to make use of this information in a meaningful way. At the very least teachers need to employ a wide range of technological tools and software as part of their own instructional repertoire.

For educators at all levels who are preparing students



for the information age, the challenges of introducing and integrating ICT into education have become even more challenging. The teacher of the future must not only be accomplished in the use of ICT but also in the integration of ICT into the curriculum. It is therefore inevitable that for any effective change to occur the use of ICT in teaching and learning must be accompanied by a corresponding change in curriculum. As Yelland (2001) has unmistakably pointed out, effective computer integration is not found in classrooms that are traditional and use didactic teaching methods.

Many countries throughout the world have launched initiatives in response to these challenges but to Nigeria it is lipservice. The integration of information technology into school curricula is almost invariably a key element of such initiatives. Numerous professional development programmes have been initiated to provide teachers and teachers-to-be with the opportunities to recognize that use of ICT can improve the quality of the educational experience by providing rich, exciting and motivating environments for learning. Research has informed these initiatives and has illustrated the benefits of ICT for student learning.

Several studies indicate that high motivation is evident in students who use ICT to learn whilst other studies highlight the opportunities which ICT presents to encourage the development of creativity, imagination and self-expression. All of this highlights the very complex nature of integrating ICT across the curriculum and the challenges that we face daily.

While as educators realize that use of ICT can be a valuable resource for improving teaching and learning, the process of integrating technology into the curriculum is not easily or quickly accomplished. It challenges all educators to reconsider teaching practice, the curriculum, the role of teachers and the ways in which ICT can be incorporated into school curriculum to maximize educational outcomes.

This paper discusses ways in which ICT will be applied in educational settings as a tool to assist and generate learning across all levels of education. It highlights the need for effective professional development to be provided for teachers so that they are able to adequately identify the educational issues involved in the integration of ICT.

Any implementation calls for substantial rethinking of curricular and educational practice. Time and effort must be expended when developing in teachers' information and values. Teachers need to go beyond training in technical competence and the use of the technology merely as a tool.

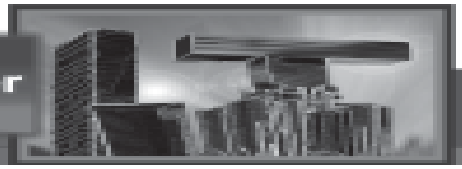
The focal point should be that use of ICT must provide students with a variety of experiences and contexts to integrate their skills and knowledge both in and out of school.

Finally, true cross-curricular integration can take place only when teachers appreciate that integration of information technology has rich potential for enabling more effective learning where the technology is subordinated to the learning goals of the school. It is only then that teachers will critically select the appropriate digital content based on the needs and learning styles of their students and infuse it into the curriculum. When this paradigm shift occurs, teachers will have become more sensitive to the kinds of learning that students engage in and can critically examine the implications for the management of learning and the effects on students' attitudes and perceptions of the task in hand. Last but not least, in discussing the use of ICT as a co-curricular resource, a working model for the development of an integrated learning environment is used to amplify all of the issues raised in the paper. It will concentrate on the overlap between IT and communication Technologies, such as use of the WWW, E-mail, Computer conferencing, video conferencing etc rather than taking the whole range of IT applications in education. Institutions including the National Open University of Nigeria (NOUN) will be able to use it to inform cross-curricular planning and the organization of resources.

WHY BOTHER USING ICT IN TEACHING AND LEARNING IN NIGERIAN SCHOOLS?

As sophisticated electronic technologies become standard equipment in schools, the roles of students and teachers will change. Before this change can take place it is critical that teachers accept the use of ICT in the classroom. Concurrently school management teams need to plan Using ICT for Teaching, Learning and Management and provide as much scaffolding as possible to ensure that teachers interpret the prospects of technological innovations in a positive manner because schools have been known to be resistant to curriculum innovations. Unless teachers are given recognition for the key role that they can play, they will continue to be gatekeepers of classroom change and prevent themselves form being able to focus on the key issues that really matter. The following are particularly important. When introducing ICT in schools, all teachers need to:

- emphasize content and pedagogy not the level of sophistication with hardware and technical skills;
- engage students in meaningful and relevant learning;



- allow students to construct knowledge;
- bring exciting curricula into the classroom;
- integrate technology into the curriculum and devise alternative ways of assessing student work;
- adapt to a variety of student learning styles to cater for individual learning differences; and
- continuously expand and welcome further opportunities for teacher learning.

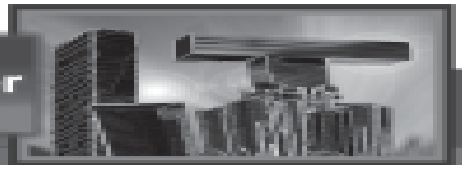
The ability to achieve all of the above implies that lecturers must have a comfortable level of ICT skills. They must acquire the basic skills in ICT and then go on to ensure that the students in their charge can also make progress in use of ICT in an incremental way. Unless teachers are functioning at a comfortable level of ICT skills and knowledge they will be unable to use ICT as a primary tool for teaching and learning across the curriculum. Even supposing that the teachers have the ICT skills they will need to carefully consider when, what and how to teach the ICT skills to their students. Teachers may choose to use very simple or complex technologies to achieve their educational vision. Bridget Somekh (1997) has demonstrated that *"the use of IT can provide innovative learning experiences, but in all cases a great deal depends upon the teacher to provide the context which makes this possible."* She goes on to indicate that *"teachers need to be competent and confident users of hardware and software, but this in itself is not enough. They need also, to understand how to organize the classroom to structure learning task so that IT resources become a necessary and integral part of learning rather than an add-on technical aid".* She asserts that unless teachers "*believe in an innovation it is very unlikely that they will introduce it effectively".* Teachers need to be convinced of the value of ICT because many teachers tend to perceive themselves to be technologically incompetent and often feel deskilled and demoralized when they first begin to use computers in the classroom. Therefore, whenever schools consider introducing ICT as a subject in its own right, it must always go hand in hand with the integrative approach.

Using ICT for Teaching, Learning and Management must be accompanied by the teacher's keen interest in expanded paradigms as they relate to the roles of teachers and students. Learners must be encouraged to construct, evaluate, manipulate, and present their ideas while demonstrating understanding of curriculum concepts and innovate constructs. It is precisely in this way that we should aim to prepare our students to work in the types of classrooms where knowledge is actively used and students are given more responsibility for their own learning when ICT is effectively integrated as a tool of technology into the curriculum.

Clearly we are all convinced that teachers need to inculcate the willingness to learn enough about ICT to make effective use of it in the classroom. For teachers to rethink and re-structure teaching and learning they must first learn enough about the relevant technologies to apply them in their professional live, and to translate them to their students as part of the integrated learning of the subject matter. Whereas in the past, the role of school was thought to be the dispenser of information, in today's context this role can no longer hold. The information explosion has changed the nature of knowing from the ability to recall information to the ability to define problems, to retrieve information selectively and to solve problems flexibly, which therefore changes the nature of learning from the need to master topics in class to the need to learn autonomously.

Teachers and students now need to learn how to learn in an ICT rich environment. Teachers need to perceive ICT as primarily a tool for teaching and learning across the curriculum although there are foundation skills in ICT that students need to learn before they can participate fully in an ICT rich classroom. The National Council for Educational Technology (NCET, 1994) in identifying potential outcomes, stated that the effective use of IT can:

- provide the flexibility to meet the individual needs and abilities of each student;
- reduce the risk of failure at school;
- provide students with immediate access to richer source materials;
- present information in new, relevant ways which help students to understand, assimilate and use it more readily;
- motivate and stimulate learning;
- enhance learning for students with special needs;
- motivate students to try out new ideas and take risks;
- encourage analytical and divergent thinking;
- encourage teachers to take a fresh look at how they teach and ways in which students learn;
- help students learn when used in well-



designed, meaningful tasks and activities; and

- offer potential for effective group work.

The fulfillment of any of the above outcomes could be extremely daunting in the eyes of the teacher. Many ICT skills are assumed and students would have to be provided with the opportunity to learn these skills in the right context. It is still the case that in many countries teachers and students continue to use computers in a presentational mode while use of ICT in the general content area curriculum is neglected or grossly underdeveloped.

IMPLICATIONS OF USING ICT IN TEACHING, LEARNING AND MANAGEMENT TO TEACHERS AND ADMINISTRATORS:

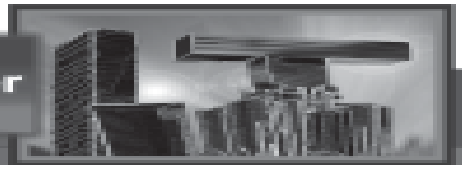
There are indications that teachers and administrators are beginning to recognize that computer skills should not be taught in isolation, and that separate *"computer classes"* do not really help students learn to apply computer skills in meaningful ways. This change is an important shift in approach and emphasis. A meaningful, unified computer literacy curriculum must be more than the teaching of isolated skills.

While specific technical skills are certainly important for students to learn, they do not provide an adequate foundation for students to transfer and apply skills from situation to situation. These curricula address the "how" of computer use, but rarely the "when" or "why". Students may learn isolated skills and tools, but they will still lack an understanding of how those various skills fit together to solve problems and compete tasks. Students need to be able to use computers flexibly, creatively and purposefully. All learners should be able to recognize what they need to accomplish, determine whether a computer will help them to do so, and then be able to use the computer as part of the process of accomplishing their task. Individual computer skills take on a new meaning when they are integrated within this type of information problem-solving process, and students develop true "ICT literacy" because they have genuinely applied various computer skills as part of the learning process. It needs to be cautioned here that from the experiences of many countries, teaching ICT as an isolated discipline is not an effective way to encourage the use of ICT in learning. This method cannot provide a meaningful environment for students to learn and apply the techniques in context. Teachers must invariably integrate ICT elements into different key learning areas and encourage students to use ICT to enhance their learning. The success of this integration depends very much on a number of factors, including the nature of the subject content and the readiness of teachers. Unquestionably it is the teachers who hold the key to the effective use of technology to improve learning. But, if teachers do not fully understand how to employ ICT effectively to promote student learning, the huge investments in ICT initiatives will easily be wasted. Notwithstanding the many years of experience in a variety of educational systems, the issue of how to introduce the use of ICT to teachers and to encourage them to use it in the classrooms has yet to be resolved one way or another in many countries. It is true to say that this issue continues to haunt many policy and decision makers today.

USING ICT AS A TOOL IN THE NIGERIAN SECONDARY SCHOOLS

The discussion above has highlighted the importance of integrating ICT into the curriculum to enhance learning outcomes. Clearly, the curriculum must be adapted or re-designed so that it is ready for ICT integration. A tool approach assumes that general-purpose software such as word processing or paint programs or an Internet World Wide Web browser, can be flexibly applied by the learner to various topics but students are still not playing an active role. And by active, we mean constructive learning. When students play an active role the role of the teacher changes to that of a facilitator of learning. The list below highlights the possible changes. Changes in classroom structure as a consequence of schools that have adopted a fully integrated technology are listed by Collins (1991) as the shift:

- from whole class to small-group instruction;
- from lecture and recitation to coaching (teacher's role from 'sage on the stage' to 'guide on the side');
- from working with better students to working with weaker students, facilitated by student-directed learning;
- toward more engaged students;
- from assessment based on test performance to assessment based on products, progress and effort.
- from a competitive to a cooperative social structure;
- from all students learning the same things to different students learning different things; and



- from the primacy of verbal thinking to the integration of visual and verbal thinking, with organizational, artistic, leadership and other skills contributing valuably to group projects.

ICT literacy will become as important as literacy in language and mathematics. Rowe (1993) asserts that *"like reading, writing and mathematics, computing gives the student a basic intellectual toolbox with innumerable areas of application. Each one of these tools gives the student a distinctive means of thinking about and representing a task, of writing his/her own thoughts down, of studying and criticizing the thoughts of others, or rethinking and revising ideas, whether they are embodied in a paragraph of English, a set of mathematical equations, the simulation of a social process, or the development of a computer programme. Students need practice and instruction in all these basic modes of expressing and communicating ideas. Mere awareness of these modes is not enough"* (Rowe, 1993: 71). It is worth pointing out that in educational settings where computers are used, they should not themselves be the primary objects of study, but essentially a tool. Rowe has contended that research conducted into *"the effects of the computer on students' cognitive development has too often tended to regard the computer as a single factor of change introduced into a classroom, which is presumed otherwise to remain the same".*

Computer is perceived as an independent variable the net effects of which can be controlled and quantified. In reality there are not net effects. The introduction of the computer into the classroom is far more than a treatment. The characteristics and potentialities of the computer become inextricably intertwined, not only with the way students might go about learning and problem solving tasks, but with the tasks themselves and the whole context of learning and teaching. It is not the features inherent in the tool but how students and teachers use it that determines the effects of computers in education" (Rowe, 1993:83).

EFFECT OF USING ICT ACROSS THE CURRICULUM IN NIGERIAN SCHOOLS

A working model for designing an integrated learning environment when ICT is introduced into the classroom teachers need to consciously redesign learning environments so that students can transfer their newly gained ICT skills and confidence to other applications that can be used in an ICT rich environment. Once teachers and students acquire some ICT skills they can adopt a transferable learning style so that each further development in ICT use whole become an easier step. Essentially when the learning environment has to be redesigned we are implying a far-reaching paradigm shift for teachers. Table 1 below depicts this shift.

Table 1: Paradigm shift in teacher's role

From To Objective learning theory Constructivist learning theory

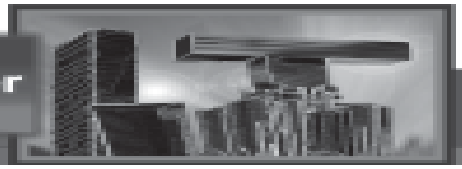
(Teacher-d Student-d)

- Teacher as expert, information giver
- Teacher as facilitator, coach, guide
- Teacher as knowledge transmitter
- Learner as knowledge constructor
- Teacher in control learner in control
- Focus on whole classroom teaching
- Focus on individual and group learning

In ICT rich environments it is more conducive for teachers to begin to help students pursue their own inquiries, making use of technologies to find, organize, and interpret information, and to become reflective and critical about information quality and sources. Teachers become advisors and facilitators of learning helping students to frame questions for productive investigation, directing them toward information and interpretive sources, helping them to judge the quality of the information they obtain, and coaching them in ways to present their findings effectively to others. This ultimately requires teachers to become even better prepared in the content of the subjects they teach, and the means by which the content can be presented. Using ICT for Teaching, Learning and Management taught and learned. In all of this, teachers need an "attitude" that is fearless in the use of ICT, encourages them to take risks, and inspires them to become lifelong learners.

IMPLICATION OF ICT AS A SUBJECT AND TOOL IN THE NIGERIAN UNIVERSITIES

Recent reforms in the ICT education and challenging ICT demands in the Universities and Distance Education in the neighbouring countries will contribute to a remarkable shift in the way ICT businesses are organised. The ICT higher education will play a major role in fostering the ICT business through establishing linkages to various Universities and enhancing collaboration to better serve the sectors needs and leapfrog into a knowledge-based economy. It will be a national effort, the academic entrepreneurship centre



of excellence has to be established in each University especially in the areas of school of Engineering and Technology in an effort to promote creativity and innovation in the field of applied ICTs, and more importantly, crystallize business, faculty/school. And student intellectual ideas and transforming it into real life products.

The centre will establish a number of industry incubators related to national and international ICT companies such as system software Computer and Communication Systems, Mobilecom and Integrated Technology group.

The centre will cooperate with the industry incubators to develop local/Nigerian software products in a variety of applications including medical and pharmaceutical applications, engineering sciences, system designs, business strategies and solutions, and complex system modeling and simulation. The centre will also offer opportunities to students and faculty to work on projects that could potentially provide challenging learning experiences and meet market needs.

The incubators might decide to accept about fifty student internship each year and for a period of six months. Each trainee shall work full time for incubators with a minimum of eight hours a day.

The trainees will work jointly with the incubator professionals and will be trained on leading edge ICT technologies. The incubators will help students enhance their personal competency, research, and skills trough practical training in problem solving and participation in project-based learning activities. Students will better understand workplace interactions and gain professional, technical and teamwork skills. The incubators will also increase student knowledge and awareness about latest developments in the ICT-related fields and enhance their creativity, innovation and entrepreneurship spirits. Furthermore incubators will offer career opportunities in variety of disciplines to graduates who prove to gain professionalism and high profile qualifications and experiences.

MOTIVATIONS:

If the centre of excellence is formed in each university, there is need to form an industrial advisory board from varieties of industries around Nigeria. The duty of the board is to facilitate and coordinate universities – industries interactions and collaborations with respect to internships, research and student projects. A number of under graduate projects and graduate research thesis will be determined with these companies. This, however, will require administration and follow up overhead as more projects and thesis are solicited and determined. Moreover, a number of promising student projects with potential market will be developed each year. Students will work on projects such as academic portals, software automation biomechanics, multimedia, database applications and much more. These projects will require special packaging documentation, and standardization to become real-life products. The motivation behind creating such a center is to account for Nigerian-industry interaction overhead and to allow students work in a business environment.

OBJECTIVES:

The Academic Entrepreneurship Centre of Excellence (AECE) if established will have a mission to:

(i) Facilitate collaboration between academia and industry to better serve ICT (Information and Communication Technology) sector needs and contribute t the national shift towards a stronger knowledge economy,

(ii) Develop a working environment for professors and students to better enhance understanding of workplace interactions, team work, and technical skills, and to increase knowledge and awareness about latest developments in the ICT – related fields.

(iii) Promote and encourage ICT companies to establish their own organizational units within the center for Research and Development, and

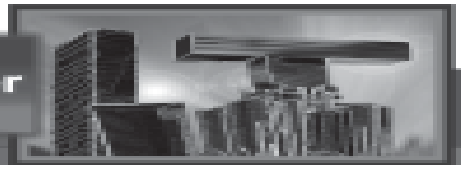
(iv) Build capacity and expertise to develop and deploy top-quality software solutions according to international and best practices standard.

THE FATE OF IT IN THE NATIONAL OPEN UNIVERSITY

Nigeria has taken a bold step in establishing the National Open University of Nigeria (NOUN) to represent distant education at University level with laudable blue print on IT implementation yet undeniably IT standardization is at a critical crossroad. How do we fix this situation to normalcy? Although the National Open University of Nigeria is affiliated to four Open Universities abroad,

1. British Open University
2. South Africa Open University
3. Indira Ghandi Open University and
4. Australia Open University

where some of their programs are adapted, they have linkages although some courses are adjusted to Nigerian



environment, the position of IT in the National Open University of Nigeria is still the traditional method which is old fashioned. Students of national Open University come for registration after payment and collect their course materials at their various study centres and go back to their various stations. Each study centre has a facilitator drawn from other universities but no formal lectures are had between the facilitator and the students. Students are summoned for tutorials once or twice a semester. A counselor is at each study center to handle students problem both academic and otherwise.

A student goes back to study the course materials and comes for examination. There are no interaction between the student and the lecturers in terms of IT. Students are not expected to read Computer Science without a framework well planned and the practical involved and is expected to have the know how of IT; these students end up as half baked in terms of IT. The affiliated open universities of other countries have a well planned core curriculum framework which they put on the internet and distant learning students use e-mail to interact with their lecturers. In the National Open University of Nigeria these facilities are far fetched.

There is great need for the design of a core curriculum framework for the Nation Open University of Nigeria.

CURRICULUM RESTRUCTURING IN THE NIGERIAN UNIVERSITIES / NATIONAL OPEN UNIVERSITY OF NIGERIA

The move to outcomes-based education and a centralized curriculum framework in National Open University of Nigeria (NOUN) is indicative of national and global trends. This curriculum restructuring is part of a broader trend to educational restructuring whereby schools are largely responsible and accountable for their own management. Educational restructuring in National Open University of Nigeria has been shaped by the implementation of such policies as *Focus on Schools, The Leading Schools {Program* and now the implementation of syllabuses in key learning areas. The syllabuses will be developed by the National University Commission (NUC) is expected to be progressively implemented over the next five years. This restructuring can also be used by other tertiary institutions.

At one level, a centralized curriculum framework appears to be in tension with Education National Open University of Nigeria restructuring policies promoting school based decision making and management, and in turn these contradictory tendencies can create dilemmas for lecturers. On the one hand, the framework can be construed as providing scope for school based curriculum decision making and development, and so be consonant with these policies. Through insights gained from a research work on school curriculum leaders, it became apparent to me that these curriculum leaders were concerned not so much with contradictory tendencies between restructuring policies as the challenges the new framework posed for lecturers, and the implications of these challenges for the intensification of lecturers work (Hargreaves 1994). Challenges related to: an understanding of the meaning of outcomes-based education; a redefinition of planning approaches, teaching methods and assessment procedures; the emotional demands of change; organizational reconstruction; adequate resource provision.

The focus of my research therefore moved to an investigation of the implementation of the IT syllabuses through school cluster planning and consideration of the implications for lecturers' work.

Cluster planning is in its formative stages and the paper will report on developments and research findings so far. Against this background, the paper will critique the idea of school cluster planning and consider its significance for curriculum restructuring and the circumstances of teaching.

WORKING AS A CLUSTER

Major factors held to influence feelings, either positively or negatively were:

- past experiences of change
- knowledge of syllabuses and planning procedures
- anticipated support and professional development
- presentation and organization of syllabuses
- likely impact on work

Lecturers identified a number of barriers to implementation and also described factors that would promote successful change. Main categories and their constitutive elements to emerge from the analysis are presented in Tables 1 and 2.

A significant majority of respondents (N=131) were in favour of school cluster planning. While 144 lecturers believed it was important for them to be involved in school program planning, a smaller number (98) wished to participate in planning at the cluster level.

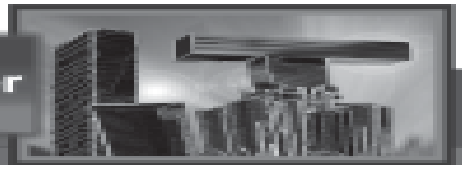


Table 1 Major Barriers to Implementation

KNOWLEDGE
- open-ended nature of the syllabuses
- ack of understanding of outcomes based concepts
- lack of understanding of implementation procedures.

ORGANISATION
- establishing a committee to organize development of the programs
- lack of communication and collaboration
- developing a cohesive school program
- staff and year level changes

SUBJECTIVE/EMOTIONAL REALITIES
- feeling deskilled
- sustaining energy and motivation
- workload
- stress

RESOURCES
- insufficient resources, support materials
- time to develop understanding and confidence to plan; to allocate sufficient attention to all

PROFESSIONAL DEVELOPMENT
- Inequitable inservice

Table 2 Major Factors Facilitating Implementation

VISION
- Clear, agreed expectations with total staff involvement.

SCHOOL ORGANISATION
- opportunities to network pre, during and post implementation
- success sharing
- communication

CURRICULUM ORGANISATION
- Clear, user friendly plans
- Guidelines for assessment and reporting
- Year level discussions and planning

PROFESSIONAL DEVELOPMENT
- Sound inservice, workshops in school time
- Workshops to discuss concerns and misunderstandings, implement, back to inservice to discuss problems
- Professional assistance in class
- Ongoing support

RESOURCES
- Time for planning, trialling
- Support materials

Lecturers expressed the view that involvement at the school level can

- Enhance understanding and expertise
- Promote commitment, involvement, and ownership
- Build confidence

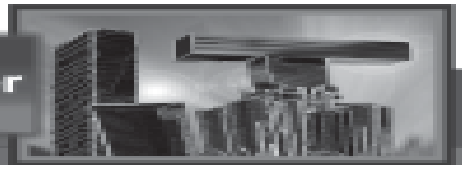
Lecturers were of the opinion that involvement at the cluster level provides opportunities to

- Use time efficiently
- Share workloads, expertise and resources
- Gain insights into how lecturers in other settings interpret and adapt the syllabuses.

The adapted framework illustrates key elements of successful school achievement as they relate to the circumstances of teaching. In other words, the framework presents these key elements from the perspective of lecturers' messages about workplace changes that need to occur in order to successfully implement the new curriculum. In recognizing the circumstances of teaching, the adapted framework adds another dimension to the process of achieving successful school outcomes.

THE SIGNIFICANCE OF SCHOOL CLUSTER PLANNING FOR SUCCESSFUL SYLLABUS IMPLEMENTATION

The proposed National Open University of Nigeria curriculum framework for the key learning area syllabuses is based on the idea of outcomes-based education. Spady (1993) identified three forms of outcomes-based education: *traditional, transitional and transformational.* In *Traditional* outcomes-based education, curriculum content and structure remain the same but the focus moves to outcomes as related to skills and competencies. *Traditional* outcomes-based education is concerned with students' success in schools and 'rarely challenges the nature of the school day, for example the time-defined structure of curriculum content'. *Transitional* outcomes-based education is underpinned by a different conception of an outcome. The outcome defines 'what is most essential for our students to know and be like in order to be successful once they have graduated'. As transitional outcomes-



based education is focused on students' culminating capabilities, curriculum and assessment are organized around higher order exit outcomes. Higher order competencies such as critical thinking, problem solving and effective communication are emphasized and 'subject matter becomes more a vehicle to assist in the cultivation and integration of higher order competencies'. *Transformational* outcomes-based education is future oriented in that it is concerned with 'the broad role performance capabilities of young people and their ability to do complex tasks in real settings, in real situations, relating more directly to life'.

In certain respects, the curriculum framework proposed for National Open University of Nigeria can be aligned with Spady's conception of transitional outcomes-based education. The framework identifies a capacity for lifelong learning as a culminating outcome, and provides opportunities for students to develop the attributes of lifelong learners.

The National Open University of Nigeria curriculum is designed to assist students to become lifelong learners. The overall learning outcomes of the curriculum contain elements common to all key learning areas and collectively describe the valued attributes of a life long learner.

PROFESSIONAL SUPPORTS

- Equitable inservices
- Inservice in school time
- Workshops to discuss concerns and misunderstandings, implement, then back to inservice to discuss problems
- Ongoing support
- Professional assistance in class

The curriculum framework is underpinned by a learner-centered approach to learning and teaching where learning is regarded as 'the active construction of meaning, and teaching as the act of guiding and facilitating learning'. The approach sees knowledge 'as being ever-changing and built on prior experience'. Assessment focuses on students' demonstrations of learning outcomes, and is held to be effective and efficient 'when embedded within a learning and teaching sequence'.

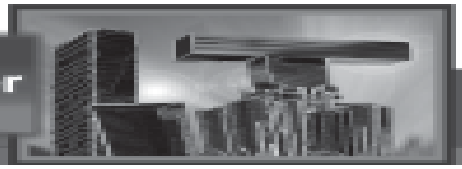
Major concepts for each of the key learning areas are organized into strands while level statements summarize learning outcomes at each level. These statements provide the conceptual framework for developing core and discretionary outcomes. Core and discretionary learning outcomes for each strand are presented in increasing order of complexity. Core learning outcomes describe learnings considered essential for all students and describe what students know and can do. Discretionary outcomes describe what students know and can do beyond what is considered essential at a particular level.

The concept of transitional outcomes-based education as exemplified in the proposed National Open University of Nigeria curriculum has implications for the approach to curriculum planning at the school and classroom level, the way teachers teach, the way students learn, and the way students are assessed. Traditionally in National Open University of Nigeria curriculum planning has been guided by educational objectives formulated according to content (that is, knowledge, skills, values, attitudes) to be learned. Content has been delivered largely though a transmission teaching style. Pen and paper tasks have been the predominant method for assessing student learning.

An outcomes-based approach to curriculum planning relies on a process of designing down (Spady 1993) or backward mapping where the exit or culminating outcomes are the starting point for making curriculum decisions about what students learn and how students can demonstrate their learning. The following set of procedural questions (Table 3) provides one example of backward mapping as can be applied to outcomes statements at the various levels of the syllabuses of the National University of Nigeria curriculum framework. The questions give some insight into the ways in which lecturers have to rethink traditional approaches to planning, teaching and assessment.

Implementation of the National Open University of Nigeria curriculum framework has significant implications for lecturers work. Adopting an outcomes based approach will entail a reconstruction of professional knowledge, in particular a redefinition of planning procedures, teaching approaches and assessment practices. The nature of these changes are well captured in the following observation by Griffing (1998).

> *The role of the lecturer must change. The role of assessment must change. The mode of curriculum delivery must change. The role of the lecturer needs to change from the transmitter of information to a facilitator of learning. Assessment needs to focus on progress along predetermined continua of learning and changes in the learner. Curriculum needs to*



maximize the students' opportunities to establish an enquiry approach to learning and to use a range of resources to lead the student along the most appropriate learning pathway to achieve the designated outcomes.

From his investigations of the implementation of profiles and outcomes-based education 'were positive about the use of profiles', Griffing concluded that while teachers and administrators used the language of outcomes-based education (OBE) and profiles. 'OBE is not widely understood at the classroom level'. He believed professional development was a critical factor and identified a range of focus areas:

- Inquiry learning and student involvement;
- Integrating the roles of lecturers, learners and administrators in profiles implementation;
- The role of the Deans/Vice-chancellors in implementation of profiles;
- Selecting consultants to aid in implementation of profiles;

Table 3

Procedural Steps For Outcomes Based Planning

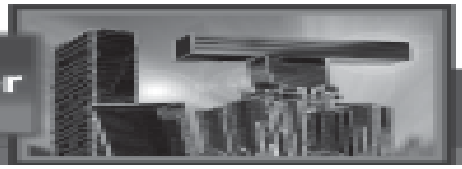
1. What will students be?
2. What will students do?
3. What knowledge/understandings, skills, values, do students need to develop?
4. How can lecturers scaffold instruction and organize learning opportunities in ways that enable students to actively construct these understandings, skills, and values?
5. How will students demonstrate what they know and can do?

- Outcomes-based education: its nature and demands on schools;
- Criterion-referenced interpretation of assessment information;
- Profiles and portfolios;
- Introducing cultural change in schools to assist in outcomes-based education;
- Use of resources in outcomes-based education;
- Professional development for lecturer educators;
- Equity issues and outcomes (Griffin 1998, p. 19).

Brady (1996) also recognised the magnitude of change associated with a move to outcomes-based education, and, following reviews of literature on outcomes-based education and teacher change, provided four major recommendations to implementers. Like Griffin, Brady underscored the importance of professional development in relation to each of the recommendations. In brief, the recommendations state:

- There is a need for a full understanding of the substantive nature of the change, what that change means, and the nature of change in general.
- There is a need to provide skills n implementing outcomes-based education, understanding that it entails new ways of developing curriculum in relation to content, method and assessment procedures.
- There is a need for ongoing support from State departments and regions after initiation to ensure effective implementation, achieving a balance between pressure to implement, and high quality assistance.
- There is a need to more fully take account of teacher attitudes and school cultures in implementing outcomes-based education.

When the sets of factors identified by Griffin (1998) and Brady (1996) are compared with those identified by teachers as represented in Figure 1, certain similarities are apparent. Lecturers noted the importance of developing a shared understanding of the substantive nature of the new curriculum framework and its implications for their curriculum planning and practice. They also emphasized the need for professional development, various forms of ongoing support, and a collaborative work culture.



Reculturing changes the conditions of lecturers' work in that it entails some fundamental shifts as illustrated below:

From individualism to professional community: Lecturers are replacing the individualism, isolation and privacy of traditional classrooms in favor of new norms of collegiality, openness and trust.

From teaching at the center to learning at the center: The shift from 'What do I do as a lecturer and transmitter of knowledge?' to 'How can I plan with others for what students do as learners?' means that student work determines the agenda for teacher work.

From technical work to inquiry: systematic inquiry, research, and reflection are at the core of lecturer's work.

From control to accountability: Instead of working as individuals to establish standards of behaviour, lecturers work together as colleagues to develop standards of learning to which they hold themselves and their students accountable.

From managed work to leadership: As leaders in heir classrooms, teachers relinquish 'power over' their students in return for 'power to' affect improved student performance. Outside of their classrooms, lecturers assume leadership roles as well. They gain responsibility in areas traditionally reserved for administrators – instruction, assessment, rules, procedures, and governance (adapted from Miller 1998, pp. 530-531).

CONCLUSIONS

Any developmental work with ICT must be designed to undertake careful foundation work in a systemic and systematic approach.

Using ICT for Teaching, Learning and Management underestimated if the policy and decision makers truly wish to go beyond surface change that will be washed away with the next demand on the educational system.

In Davis (2001) it has been suggested that the power stations will be the teachers as they rise to the challenge of supporting individual learners and their own professional development needs. Policy makers worldwide will want to create such a community of practice where the teaching profession starts to promote its own development and that of the educational system.

In time all countries will succeed in their endeavours to develop models of ICT use in schools that can facilitate lecturer coping with ICT integration so as to set students on a path to lifelong learning. Persistent efforts can achieve a flowering of innovative teaching and learning that will result from a change in lecturers' beliefs as well as practices.

This paper has argued that successful implementation of the curriculum relies on lecturers learning about different approaches to curriculum planning, teaching, and assessment, and this in turn involves a re-culturing of lecturers' work. The process of re-culturing recognizes that lecturers, like their students, learn through the active construction of meaning, and the paper has advocated that Nigerian schools, universities, National Open University and tertiary institutions will not be left out in the reforms where there is an emphasis on teaching for understanding. There is great need for policy makers to shift efforts from designing controls to developing capacity.

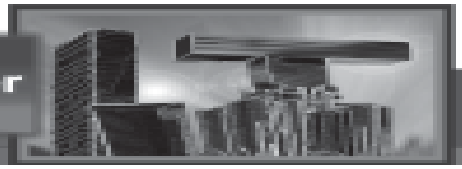
References

Collins (1991). The role of computer in restructuring schools. Phi Delta Kappan. Cited in Dillemans, R. Lowyck, J Van der Perre, G., Claeys, C., & Elen, J. (2000) New Technologies for Learning: Contribution of ICT to Innovation in Education, P. 75

Davis, N E (2001). The Virtual Community of Teachers: 'power Stations' for learners nationwide? In M. Leask(Ed.) Using information and communication technologies in schools: Key issues. London: Routledge.

Dede, C (2000). Emerging influences of information technology on school curriculum studies, 32(2), 281-303. Rowe, H 91993) Learning with Personal Computers, ACER, Victoria. Somekh B and Davis N.E.(Eds.)(1997). Using IT effectively in teaching and learning: Studies in pre-service and in-service teacher education.

Hannafin, M (1999). Learning in open-ended environments: Tools and technologies for the next milennium.



IT Curricullum Review: NACOSS Position

Olusegun C. Olutayo, Olugboji Tolulope, Olusanya Gbolahan

If there is any concern we have had as a student association, then it's been no other, than the review of the curriculum of instruction in our tertiary institutions of learning. As an association Engineered by students for students, with a responsibility to address student related concerns, we have considered this issue of utmost priority. Today, therefore, it gives us great pleasure that the Nigerian Computer Society has dedicated this portion of the conference to address this concern of ours. We know that it is not only our concern, but the concern of industry as well as every well meaning Nigerian whose desire it is to see that Nigeria meets the UN Millennium development goal of E – readiness.

One need not look to far to understand why we clamour passionately for the review in the curriculum to which the Nigerian IT youths are currently being subjected to in today's Universities, polytechnics and colleges of education. It is general knowledge amongst all educators that the purpose of education, amongst other things, is to prepare its students adequately for the business of life. Therefore the first and most important criteria that identifies the relevance of the curriculum – tool of education, is how well it prepares its products to face their future career challenges.

In our case, what is the business to which we shall engage ourselves, after our eventual departure from our respective citadels of learning? Well, none other than the activities to which the ever growing and dynamic landscape of Information Technology shall requires of us. These challenges are placing a test on the quality of IT education we are receiving from the Nigerian Institutions of learning.

The impact of this reality is fully visible on the dejected faces of fresh graduates who are rejected job offers for their lack of the barest minimum competence and skills that are required of them. The obvious thus leads us to ask: Does the present Curriculum cater for the needs of the Industry? Does it service the desires of the employer in his search for competence in his prospective employees? Does it equip the future drivers of Nigeria's Information society for the business of Information Technology?

Certainly if we digress just a little and leave the terrain of smug theorising to approach the stark realities of cold hard facts, it will become abundantly clear that the answer to all these questions is an irrefutable No! There is no solace in the current system for the expectations that Nigeria requires to jump start her into the landscape of global relevance. A look at these facts justifies this verdict.

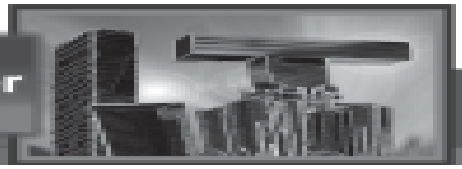
Amongst all the institutions of learning in Nigeria, the focus of computer education still remains the study of the science of computing rather than the application of computing innovation to address industry, commercial and societal demands. These demands include skill specializations in E - governance, E - commerce, E-Business, Web – Centric Learning and the other numerous requirements of the Information Technology Landscape. Even the courses taught in the computer science curriculum, are quite outdated. In Nigerian colleges programming courses like Pascal, FORTRAN and COBOL still take front stage as opposed to new and more relevant programming methodologies and Tools that are in use in more advanced countries. Getting qualitative education in the light of this reality has indeed become an arduous task.

Presently, in most cases getting a good industry relevant education necessitates that one engages in investing in substitutes like other expatriate educational institutions, which provide an alternative in the face of the incompetence of the Nigerian educational system. In most cases this is extremely expensive, and the process is often aborted for this reason. For those that are lucky to finish, they remain only but a few in the light of the other numerous many that are churned out from the system with little or no relevant knowledge and skills.

So as the representative platform for Nigerian students, we make a declaration that Change is no longer an option but an imperative that should be treated with the urgency of a National Emergency. If it is our desire to join the advanced economies of today's world, if it is our desire to replicate the Indian "miracle", then we have to tailor our curriculum to the demands of present day needs, and not be content with remain in the dark holes of self preservation.

Amongst these imperatives of Curriculum review also go the following that will enable that IT education received in Nigerian universities become relevant:

1. The Autonomy of the Universities to Design relevant courses without the overbearing influence of the Nigerian University Commission (NUC)



2. Adequate funding for training of the teaching staff which will include the Edu-Portal facilities provided by companies like Afri- Hub
3. Equipping the Nigerian Universities with infrastructure for use in Nigerian Universities: Computers, Campus – wide Networks, Subsidized Internet facilities etc.
4. Visible Collaboration between the Industry and Academia in the design of curriculum itself and in terms of funding and oversight of research projects carried out in the Universities
5. Student Incubation Training, that deals with identifying and sponsoring youth competitions with defined relevant skills for success in this competitions

So long as we continue to commit ourselves to the process and task of change, we sincerely hope, and in truth, do believe that this will mark a turning point in Nigeria's desire to emerge as a leading force in the New Economy that is Knowledge Driven.

Again the goal and target in designing a relevant model of an IT curriculum will remain, the guarantee that the products of the educational processes built on this curriculum are well equipped to face the challenges that the Hydra ended. We believe that for the Nigerian Youths the world of opportunity is open and ensuring that this youth meet this day of opportunity prepared will be the true test of Nigerian's IT education.

Nuggets for Outsourcing Information Technology Services

Longe Olumide Babatope

ABSTRACT

Outsourcing of goods and services is commonplace throughout the business-world. The latest entrant into the trend, however, is the outsourcing of information technology applications and services. This is prompted by several business factors such as the reduction in cost of employing redundant staff and equipments maintenance. However, the potential impact to security, confidentiality and integrity of organisational data, not to mention other business processes calls for some caution. For the benefits of outsourcing to be maximized at the expense of obvious risks, methods on how to identify risks with proposed outsourced agreements must be devised. This paper proposes steps to be taken when outsourcing information technology services with respect to the role of service providers. It also advises business decision makers and other professionals of the potential consequences of a right or wrong vendor selection process.

1.0 Introduction

Outsourcing has its place. It can be a cost effective method of supplementing in-house capabilities, providing additional expertise, and allowing an organization to concentrate its increasingly limited resources on those efforts, which most greatly support its strategic mission. Outsourcing is not necessarily an all-or-nothing decision. It can best be viewed as a continuum, moving from no outsourcing at all, to partial outsourcing and up to full outsourcing. An organization must decide not only whether to outsource a function (telecommunications, IT, equipments), but often which specific tasks within that function to outsource. Contrary to popular belief, outsourcing does not negate the need to manage, though it may mitigate that need to a degree.

According to Aubert (2001) outsourcing is the use of resources external to the company or organisation, which may have been previously handled internally by staff. Outsourcing of Information Technology (IT) can take shape in many forms and include a wide variety of services and support functions. Some of the most common forms of IT outsourcing include network hosting, IT jobs, source code development, and application service provision. Typically, it is support functions, which are preferential candidates for this type of business solution. Great care should be taken before considering core business competencies for outsourcing. One would even advise that organisational functions requiring managerial or executive decisions typically should not be outsourced.

1.1 Motivation for Outsourcing

Outsourcing, when executed correctly can provide many benefits to business concerns and other organisations. The reasons to outsource include reduced costs, improved performance, increased business competitiveness, access to a superior knowledge base and limited in-house staff to support the business needs. Vendors of outsourcing solutions come in a wide variety of forms and options. These can include short and long term agreements, different levels of services from basic solutions to strategic alliances. Vendors also vary in location; both domestic and international opportunities exist. Due to the very nature of information technology it is possible to select a vendor worldwide taking advantage of high-speed network connections.

With many options for an organisation it is understandable why the popularity of outsourcing is strong in today's business climate. A Gartner survey in America from early 2000, proposed that outsourcing will continue to increase locally and overseas (Insiga & Werle, 2000). This trend is also gradually becoming noticeable in the third world. For example, in Nigeria, outsourcing of IT services is becoming a regular practice

in banks, tertiary institutions and in the government sector (Olawale, 2003). However with the opportunities also arise a degree of risk and uncertainty, which could potentially impact the organisations ability to function, do business or lose market share.

1.2 Off-Shore Outsourcing

Outsourcing core business functions has been a common practice among multinational corporation in Nigeria even before the present IT wave. For example, oil companies have been known to contract software development, design of off-shore infrastructure etc to firms abroad. In contemporary times, a new terminology off-shore outsourcing, has evolved to encapsulate this concept. (that is contracting business functions to firms outside the host country). Kern et al (2002) mentioned that in the United States, corporations plan to outsource many thousands of Information Technology (IT) jobs to outside firms from 2000. Most of these jobs will belong to offshore organizations in India or Southeast Asia. As a current Information Technology professional in the new millennium, or a student considering a future career in IT, outsourcing is a business trend one must fully understand.

2.0 Changes Coming with Information Technology Outsourcing

Five or ten years ago, workers were attracted to the Information Technology field given the challenging and rewarding work, good remuneration, numerous opportunities, the promise of future growth and long term job stability (Lacity et al, 1999). Outsourcing will alter (and is already altering) each of these IT career fundamentals as stated below:

1. The nature of IT work will change dramatically with offshoring; the future may be equally rewarding, or it may prove wholly undesirable depending on one's individual interests and goals.
2. Information Technology salaries will increase in the countries that receive outsourcing contracts and will decrease in the countries outsourcing. Hence the need for us in our nation to a virile IT workforce that can take advantage of this trend.
3. Likewise, the total number of IT jobs will increase in some countries and decrease in the others. Most future growth will happen outside the U.S, and job stability will remain unclear everywhere until the outsourcing business models mature.

2.1 Coping with IT Outsourcing

IT workers in the U.S. and other advanced countries of the world may already be witnessing some impacts of IT outsourcing, but the future impacts are likely to be much greater all over the world. Outsourcing has come to stay. Our concern should be to configure our polity so as to take advantage of the gains. Tomorrow is for those who prepare for it today.

What then can be done to get ready to cope with the expected changes (Saunders, 1997). The following points should be considered:

(a) Flexibility: The prospect of job searches or career changes can be quite stressful to Information Technology workers. IT students may understandably begin to question their choice of career. However, the more stress and worry a person takes on, paradoxically the more difficult it becomes to successfully reach their career goals. Years ago in IT, specialization was the ultimate. Those with more technical backgrounds/orientation like enterprise architects, hardware engineer etc commanded the highest salaries. Nowadays, a person is much better positioned if they are skilled in multiple areas of both technology and the business side of IT. Flexibility is now king.

(b) Entrepreneurship Development

Uncertain economic tides and times of big industry change are often the best ones for starting a new business, due to lower prices for capital, less competition, and the natural emergence of big new market opportunities. All it takes is an entrepreneurial attitude and a few good ideas.

3.0 When Information Technology Is Outsourced

When IT is outsourced, there are numerous changes, which must take place. One cannot assume that one can transparently make a transition of this magnitude without significant and sweeping changes in day-to-day operations.

- Control is lost, at least to some degree.
- Costs change, but don't necessarily decrease. Non-personnel costs vary but are generally not significantly altered; outsourcers can realize some cost savings due to economies of scale, bulk pricing, etc. Personnel costs are lowered and recurring

expenses may go down if the outsourcer provides more efficient service. Overall long-term costs generally favor a prudent balance of outsourcing and in-house management and vary from one firm to the other.

- Quality may improve or suffer.
- Knowledge of organisational operation is lost for outsourced functions.
- Outsourcing doesn't eliminate management responsibilities, just changes their nature and level. In other words, someone still has to manage the outsourcers, the contract, the interface etc.

3.1 The Choice of Outsourcing

Whether to outsource or not is not a simple decision. One must take into consideration not only the function itself, but also what and how much to outsource. Evaluation of outsourcing options requires a traditional cost/risk/benefit analysis very similar to the old "lease vs. buy" question. What are the short and long term costs? Who fixes it when something breaks? How long will you own it? and what happens when the "lease" runs out?

Outsourcers/Vendors can frequently bring more and broader expertise to any given technology than is available from in-house personnel. With the continued merging of voice, data, and video technologies and infrastructures, it is hard enough to coordinate the activities of internal staff for different departments. It may become even more complicated to manage these technologies when one or an outsourcer manages more of them (Craummer, 2002). Outsourcing trends are cyclical. Various industries and functions alternate between outsourcing and in-house management. In the 80s it was very popular to outsource computing functions. Companies like IBM, EDS made millions of dollars taking over computing departments. Presently, many institutions that used to totally outsource IT have moved it back in-house, citing higher costs, lack of flexibility, and/or lack of responsiveness as the primary reasons.

3.2 The Outsourcing Model

There are generally three Outsourcing Models. These are:

Fig.1: Full Outsourcing

(a) Total outsourcing: Common for food service, custodial services, etc. Outsourced functions are seen as self contained, non-strategic, and generic. When the contract runs out, new contract is signed with another generic provider.

(b) Partial outsourcing: Common for Physical Plant, Telecommunications, etc.: The department handles in- house the day-to-day activities, basic maintenance, customer service, and other tasks, including those involving specialists and technicians. The department outsources certain jobs (or parts of jobs) that are routine, large scale, dangerous, specialized, one-of-a-kind, etc.

Fig.2: Partial Outsourcing

(c) No outsourcing (or virtually none): Admissions, registration of students, salary payment, staff promotion etc. All activities and responsibilities are handled in-house because they are unique to the system. They are part

of to the organisation's identity, marketing strategy and vital to the system's viability.

Fig.3: No Outsourcing

4.0 Decision Criteria

1. **Cost**: Can you reduce your overall costs by outsourcing? Financial modeling might provide the answer. Things to factor into the modeling process include salaries, benefits (including holidays, vacation days and sick days), training, test equipment, tools, computers, telephones, furniture, space — everything that relates to having an employee on the payroll as opposed to being on a contract and off-site.
2. **Expansion/Downsizing**: Does outsourcing make sense in terms of the organisation's stability, size and culture?
3. **Flexibility**: Can the outsourcing company provide equal or better flexibility in terms of types and levels of service, hours of operation, etc.
4. **Service/Staffing**: Does it make sense to outsource all of the services that you presently provide, or just some of them? How will you decide which ones to keep - the easiest ones, the most interesting ones, the ones with the highest profile, the most expensive ones to outsource, the ones that have a limited life span? What would be the impact on staffing levels in each case? Will the outsource company hire away some of your staff?
5. **Space**: Is space allocation, together with all associated furniture and equipment, an issue in your organization? Would it be a factor in this decision? If the vendor is off-site, how far away are they located?
6. **Control**: How much control would you lose (or perhaps gain) by outsourcing? How is security of information and university property maintained with outsourced employees?
7. **Performance**: How is performance measured? By whom?
8. **Contract and recourse**: How can you escape the contract? What is your recourse if the outsource company is not working out? Are performance penalties and/or performance bonuses addressed in the contract?
9. **Reliability**: Which arrangement would allow you and upper management to sleep easier at night? What has been the experience of other organizations that have used an outsourcing company, especially this particular outsourcing company? How does the outsourcer address disaster prevention and recovery?
10. **Company profile**: Is the outsourcer independent, or are they affiliated with a vendor? If the latter, how does this influence any decisions regarding selection of products and services for you? Is the outsourcer stable? Likely to merge? What would a merger do to your relationship?
11. **Company employees**: What are their qualifications and experience? How is performance assessment and monitoring handled? What is your recourse if an outsourcer's employee is not working out?
12. **Confidentiality:** Would it be an advantage or disadvantage to have different staff (the outsourcer's) working on your account? How about staff from several different outsourcing companies?
13. **Economies of Scale**: One attraction of outsourcing is that many of the outsourcing companies "aggregate" services from many different customers to achieve bigger discounts, and can negotiate bulk purchasing arrangements with suppliers. There may also be other economies of scale achieved by the outsourcer.

4.1 Potential Security Risks of Outsourcing

The potential for security risk varies with each outsourcing agreement. The contract and more specifically, the statement of work, contains the finer details of the agreement where security concerns should be identified (Craumer, 200~). What follows are some factors in this category

(a) Risk of unauthorized disclosure and unguided access to information: The biggest potential for risk and an innate factor with outsourcing IT services is the act of giving a third party, the outsource vendor, access to an organisations information. This can include confidential information such as financial, medical information, records and strategic development. Along with the access, is the possibility of unwanted disclosure alteration or destruction of the information. The possible impact here to the organisation is loss of reputation, competitive edge and legal risks (Insinga and Werle, 2000, Longe, 2004).

(b) Malicious attacks against outsourced software application: Code and program development through an outsourced vendor could portend serious risks for an organisation. Application service providers and Internet service providers are vendors involved in code development and program design. By the very nature of their job and the services they render it is possible for them to install trap doors within the software (measures through which software can be attacked), implant malicious codes in client's programs so as to continue to enjoy patronage and duplicate software hosted on a network for unauthorized use, enhancement and other gains. Other risks from application service providers can include the following:

- Poorly coded applications
- Access of data by other customers due to poor security features of the application
- Inadequate security policies
- In ability to enforce security policies
- Lack of incident response and disaster recovery plans
- Inadequate access control to data from the vendors employees

(c) Risks associated with remote connections

Risks associated with hosting centers and services that sometimes allow for remote access into the client networks by the vendor. Remote connections are usually established to handle large data transfer or remote administration. A Virtual private Network (VPN) or other secure channels may be established into the clients network in order to secure data transfer. This could pose a risk to the network connecting the client and vendor because intruders could capitalize on such openings to gain access to communication and information on the network. There are many examples of security issues in recent years, which had occurred during outsourcing arrangements with less than reputable vendors. In some cases, even with reliable vendors, the vendor's employees acted inappropriately in the handling of the data entrusted to them (Kern et al, 2002, Longe, 2004)

A Federal Bureau of Intelligence (FBI) case in 2002 illustrates this type of risk. In an overseas outsource venture for the purposes of debugging source code. An American company was apparently the target of an intellectual property theft scheme when the vendors employee attempted to sell the source code to a competitor which reported it to the bureau (F.B.I , 2002). This case also sheds light on other potential risks, such as difficulties in dealing with different legal systems, different attitudes towards intellectual property and the ability or priority of the foreign hosts law enforcement agencies to assist in these types of issues should the risk become a reality.

4.2 Identifying Security Risks

The potential impacts to security with an outsourcing agreement depend upon the details of the arrangement. The primary question that must be asked by the security professional is: "what are threats?, what are the impacts of such threats? and what is the frequency of these threats"? (Lacity, 1999). Some risks are easily identifiable, other may not be so apparent or even worse and later discovered after contractual agreements have been made. A great tool for identifying risk is to refer to the contract itself. The statement of work should give clear and concise language of what actions is necessary and permitted by the vendor. The service level agreement defines the acceptable levels of these activities (Baker, 1998).

4.3 Mitigating Security Risks:

Many of the risks with an outsourcing arrangement may be apparent to security professionals, but not necessarily the dealmakers within the organisation who are seeking to obtain the services of a vendor. There are several steps that can be taken to mitigate these risks, however keep in mind that it is important to understand the role of security within your own organisation first.

If an organisation does not consider security an

important area within its structure, then arguments concerning security risk may go unheard.

Methods for Mitigating Outsourcing Security Risks

Some of the methods of mitigating security risk, which should be considered, are suggested below.

(a) Involvement of Risk Management Personnel in IT projects and outsourcing: Internal communications and participation of security in IT is an important first step of understanding how the technology environment may be changing. If an organisation involves Information security staff in the process of vendor selection, preferably prior to any final agreements, then the ability to improve security can be more easily made.

(b) Choosing the Right Vendor: In addition to being involved another way to mitigate security risk is offering to review the vendor when the organisation is in the selection phase, this can be in the form of a due diligence. This might include site visit, remote security testing of vendor equipment and processes, vendor personnel practices, vendor disaster recovery practices etc.

(c) Clear and Concise Legal Definitions: The contract with the vendor should be clear as to services being provided. The statement of work should include security needs of the arrangement and the service level agreement should define the expectable levels of security. For example thresholds for notification of suspected attacks at the vendors site impacting the Organisations data. The ability to perform further reviews such as audits should be included in outsourcing agreement for periodic reviews (Baker, 1998).

(d) Contingency Plan: Simply put, organisations should have an escape route should the vendor not be able to fulfill its obligations. It is advisable to create and document plans for an unplanned exit from the vendor. Careful consideration to securing and recovering data should be thought out. Also recreation of the environment including security features should be well documented.

5.0 Conclusion

The popularity of outsourcing is increasing through various marketing activities. However with the benefits of outsourcing also comes the potential for some very real risks, which can have direct impact on information security and other processes within an organisation. We conclude by enjoining organisations that are actively planning to partake in outsourcing to examine closely methods that have been proffered to identify risk and mitigate those risks.

REFERENCES

Aubert B., Rivard, S. and Patry, M. (2001): "Managing IT Outsourcing Risk: Lessons Learned," CIRANO, Montreal.

Baker, R. (1998): "Articulating a Litigation of Outsourcing Strategies". Available online at http://www.brco.com/articles

Bates M. Davis, K. and Haynes, D.(2003): "Reinventing IT Services," The McKinsey Quarterly, Vol.1, No. 2.

Craumer M. (2002): "How to Think Strategically About Outsourcing." Harvard Management Update, Harvard Business School Publishing,

Compass Consulting (2004): "Outsourcing - Is It Right For Higher Education Technology Services?" Compass Consulting International, Inc.. New York.

F.B.I (2002): "New Era of Cyber-crime Co-operation". Available online at http//:www.fbi gov/Daoe2/bosind. htm

Insinga R.C. and M.J. Werle (2000): "Linking Outsourcing to Business Strategy," Academy of Management Executive, Vol.14, No. 4.

Kern T., Willcocks, P and Van-Heck, F. (2002) "The Winner's Curse in IT. Outsourcing:

Strategies for Avoiding Relational Trauma," California Management Review, Winter, Vol. 44, No. 2.

Longe F.A (2004): "The Design of An Information-Society Based Model For the Analysis of Risks and Stresses Associated Risks Associated with Information Technology Applications". Unpublished M.Tech. Thesis, Federal University of Technology, Akure, Nigeria.

Lacity, M., Wilcox, L. and Fenny, F.(1999) "IT Outsourcing, Maximise Flexibility and Control," Harvard Business Review on the Value of IT, Harvard Business School Publishing, Harvard.

Olawale Akiniyi (2003): "Outsourcing in the Nigerian Banking Sector: Gains and Challenges". Business Times, 5th February.

Saunders C., Gebelt, M. and Qing, H. (1997): "Achieving Success in Information Systems Outsourcing". California Management Review Series. Vol. 39, No. 2.

Outsourcing of Software Components – Opportunities for Developing Countries

Bamigbola, O.M; Daramola, J.O.

ABSTRACT

In recent times, outsourcing of software development processes by bigger companies, mostly in the developed countries to the smaller companies of less developed economies has become the order of the day. The rise in globalization, and the advent of Internet, where virtually every kind of information is made available are major catalysts. Outsourcing is the process of placing the development of certain software tasks in a software development project in another development organization. The general trend in these arrangements, in most cases, is for an organization to place the outsourcing subcontract in places where it could get high competence at a reduced cost. This could be in terms of cost of labour or other development costs. The Asian countries and most notably India have benefited tremendously from these arrangements. We also, believe that Africa and most notably Nigeria can also draw from this software boom. We present in this paper, the issues involved in the outsourcing and also provide recipes for strategic positioning for outsourcing services patronage by corporate organizations in the Nigerian nation.

1. Introduction

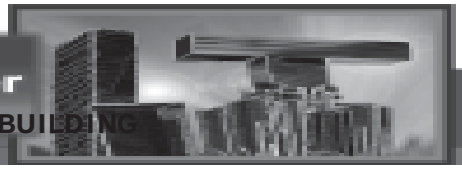
Outsourcing entails the delegation of tasks or jobs from internal production to an external entity such as subcontractor. It is the process of placing the development of system components in another development organization with the aim of achieving cost reduction and a gain in the time of development. It has been established that when companies engage in IT development outsourcing they are able to reduce development expenditure by as much as between 30-50 % [4].

Business processes can be outsourced to either domestic subcontractors or offshore subcontractors who are otherwise known as Outsource Service Providers (OSP). The rise of globalization is making many more companies in the developed countries to embrace offshore outsourcing. This entails the subcontracting of business processes to other countries in other regions of the world. Most of these jobs are presently outsourced to India (notably Bangalore), China, South East Asia, and recently Russia, Eastern Europe, Philipines and some former soviet republics, where there exists cheap labour with high professional competence [8]. The technical and professional proficiency that exist in these countries compare favourably with what obtains in the developed countries like U.S.A at a relatively cheaper cost, which makes the temptation to outsource irresistible.

These emerging trend also presents ample opportunities for other developing nations of the world especially in Africa and most notably Nigeria, because of her huge manpower endowments and her perceived potential capacity to emerge as a formidable outsource service provider nation from within the African continent. Statistics as at today, shows that Africa still occupies the back seat in the global drive for a stake in the IT outsourcing market.

2. Outsourcing : An Overview

In earlier times, the nature of software tasks was monolithic, bestowing the task of hardware design and implementation, the construction of operating system and development of application software on a single establishment. This was due to the lack of standardization in the design of the computer hardware and operating systems. This also made it difficult for small companies to compete in the software development market making only large software companies to enjoy a monopoly. However, the standardization of computer hardware platforms and operating systems, which led to the emergence of the PC technology in the early eighties, also created a level playing ground for all categories of companies and economies to become active in the software development market [3].



One noticeable trend, although more of a paradox, is that while the hardware and software development tools are getting cheaper, the nature of software development processes are becoming more expensive as a result of the growing and seemingly insatiable requirements of users. Hence, the search for means to reduce the time and cost of development, without compromising system functionalities by developers.

The search for the so called silver-bullet [1] has lead to emergence of many noble paradigms and approaches to the software development process all with the objective of producing software products that are accurate, reliable and cost effective. Some of these paradigms range from functional development, (based on modularization), object-oriented software engineering, agile approaches (Extreme programming etc.), Component-based software engineering, aspect-oriented programming and so on, leaving the developer with the choice of adopting the most suitable approach for his particular situation [12], [9], [13] [14].

The trend of globalization and the Internet has initiated a massive drive toward offshore development (outsourcing) which is the practice of moving development and other IT work such as project analysis, coding, testing and maintenance in developed countries to software companies in the low-wage areas. One software development approach that makes itself amenable to this kind of situation is component-based software engineering (CBSE).

Component-based software engineering is a new paradigm for software development that is gaining increasing relevance both in the industry and the academia. The relevance stems from the opportunity it offers software developers to model software systems at a level of abstraction higher than the object level. The notion is that software intensive systems can be built by the composition of various reusable parts (components). These components are able to integrate and interact with one another at run time in order to achieve the requirements of the overall system. Therefore, the character of component-based systems derives from the combined nature of the components' behaviour and interactions. It is also concerned with the means for the improvement, replacement and customization of software components [2], [3].

A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by a third party [13]. Therefore acquiring a proficiency in the use of modern trends of software engineering like CBSE, will no doubt boost the potentials of any establishment as an offshore outsource service provider.

There are two general approaches that are mostly used for offshore outsourcing. One is the setting up of offshore branches or annexes in the chosen foreign country, the second is the selection of certified outsource service providers (OSP) in the foreign country. Nearly all the fortune 500 companies use offshore development services or have their own branches set up offshore. Infact major hardware and software giants like Microsoft, IBM, Apple, Dell, Novel, Sun and Siemens have extensive offshore operations [8].

The seemingly irresistible appeal for outsourcing stems from a number of reasons, some of these are:

1. It has become obvious in the developed countries that a lot of money could be saved on staff emoluments, equipment, training and maintenance when projects are outsourced;
2. There is a shortage of IT professional in the developed countries because of the rate of technological development which has reached a critical proportion. There are 340,000 vacant IT positions in the United States of America alone. [8];
3. Some of the countries to which jobs are being outsourced have a more liberal business environment with lesser taxes, less stringent labour laws etc. compared to the developed countries;
4. Lastly, because technical competence can be acquired offshore at a cheaper price.

Moreover, statistical projections show that the practice of outsourcing is not about to stop soon. Data from IDC and Meta Group shows that 60% of midsized companies outsource their high revenue (HR) functions, including related IT systems and this is expected to further increase. The Forester Research Consulting firm has predicted that in the next 15 years, 3.3 million U.S. service industry jobs and US$136 billion in wages will move offshore to countries such as India, Russia, China and Philippines. [5]

2.1 Benefits of Outsourcing

Outsourcing is like a two-sided coin that could presents a win-win scenario for the partners involved and on the other hand can lead to losses and regrets if certain issues are not well managed.

2.1.1 The Offshore Outsource Service Provider (OSP)

The benefits that an outsource service provider can derive from an outsourcing subcontract includes:

1. Profitable bargain from signed outsourcing contracts leading to more profit.
2. Acquisition of more software development experience through collaboration
3. More profit for the company, which translates to higher potential for capacity building staff welfare, better recruitments etc.
4. The Offshore country benefits from its outsource service providers paying more in taxes, wealth creation, and leading to general boost in the economy.

Disadvantages:

1. There is always a huge expectation for performance which atimes may be too enormous to attain
2. Circumstance may develop that can lead to abrupt termination of contracts which may result in losses.

2.1.2 The Foreign Contracting Company

The contracting company also stands to benefit immensely from the outsourcing arrangements in a number of ways, these include [5]:

1. Acquisition of cheap technical competence with high productivity. For example a company can hire an engineer in India for US$ 10,000 a year whereas it needs between $60,000- $90,000 to do same in the U.S [15]. Tables 2.1 - 2.4 shows some statistics pertaining to some Outsourcing areas of the world.
2. Faster development is achieved with quality standard.
3. There is a reduction in overall development cost which may be up to 50% of the domestic cost. This is because of the cost of re-equipping, training of staff etc. will be removed.
4. There is improvement in the start-up time, because it is much easier and faster to start-up a project offshore that in-house, because most of the offshore companies have developed core competencies in some kinds of jobs and have a structure that enables them to take projects at any stage.
5. The retention of project management control despite outsourcing is an advantage, once definite schedules and deadlines for deliveries are set and agreed on.
6. Protection of all intellectual property rights, as it all belongs to the original company [6].
7. It also offers financial protection because payments are made at definite agreed points that milestones have been achieved.

Disadvantages [5]:

1. There is a lot of risk involved in outsourcing and so the outcome may not always be positive.
2. If proper deadlines and schedules are not put in place, the project may not progress as expected because offshore service providers have the tendency to want to give priority to the more profitable jobs.
3. It may result in loss of jobs or redundancy of domestic staff. This is still a boiling labour issue in some of the developed countries [9].
4. Cultural difference can cause problems.
5. Communication problems due to language barrier, and time zone differences in some cases.
6. Incomplete project managenemt control due to distance separation.
7. Security issues.
8. Since the development was not done in-house, a detailed system audit is necessary, which could turn out to be very expensive [5].

3. Modalities for Strategic Positioning

3.1 The Indian Example

India today, is a nation in rapid development. India's economy stands as at today as the fourth largest in the world. Its economy recorded the second-fastest annual growth rate at around 8% in 2003 and 7.5% in the first quarter (April-June) of year 2004. India's per capita income is about US$ 2,540. India's foreign reserves amount to over US$ 120 billion. The exports of goods and services of India's IT and outsourcing industries is 7 billion British Pounds in the 2003-2004 fiscal year. The overseas outsourcing revenue of Bangalore is 32% of India's total overseas outsourcing revenue of 10 billion Euros [11]. See Figure 3.1 on some statistics on India.

These achievements by India and some other countries as presented by available statistics have been a result of careful and strategic planning and positioning from the national to the corporate levels in these countries. A careful study of the policies and procedures in some of these countries is hereby put together as a recipe for the Nigeria model.

Table 2.1: Source [10]

IT labour shortage developed countries Vs IT resources available in Eastern Europe and India	
U.S.A.	800,000 (by ITAA)
EU	400,000 (by Giga Information Group
Japan	100,000 (estimation)
Total	**~ 1,300,000**
EU candidates	150,000-2000,000 (estimation)
Russia	250,000-350,000 (estimation)
Other CIS Countries	100,000 – 150,000 (estimation)
India	520,000 (by Nasscom, 2001)
Total	**~ 1,000,000 – 1,300,000**

Table 2.2 Source [10]

Outsourcing Rates in Outsourcing Areas compared with the U.S.A and Ireland	
U.S.A	$80 / hour
Ireland	$40 / hour
Outsourcing Areas	
Poland, Czech Rep., Hungary, Slovenia, Croatia, Slovakia	$15 -30 / hour
Russia	$12 – 25 / hour
Bulgaria, Romania, Baltic states	$8 – 25 / hour
Belarus, Ukraine, Yugoslavia, Central Asia	$6 – 25 / hour
India	$20 / hour

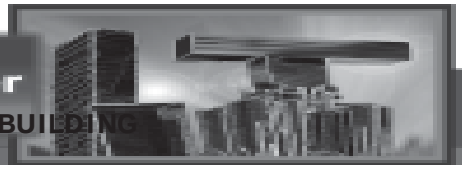


Table 2.3 Source [10]

Software Market Report 2001: Programmer's Monthly Salary Survey in Eastern Europe		
Country	**Number of Companies**	**Programmer Salary Per Month in US Dollars**
Belarus	100 – 200	300 --700
Bulgaria	300 – 500	300 – 900
Czech Rep.	200 – 300	700 – 2500
Hungary	-	800 – 2600
Romania	300 – 600	300 – 800
Russia	1000 – 1500	300 – 1600
Ukraine	800	200 - 1200

Table 2.4 Source [10]

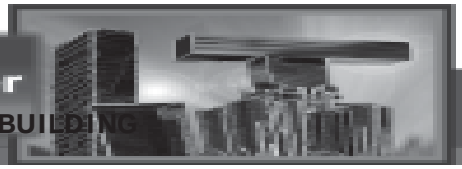
Brain Bench Certification report 2001, Approx. 600,000 tests in 30 IT categories	
U.S.A.	194211
India	145517
Russia	68050
Ukraine	23349
Romania	16122
Bulgaria	8844
Belarus	5481
Latvia	4597
Estonia	2054
Kazakhstan	1471
Czech Rep.	1460
Croatia	1070
Yugoslavia	657
Hungary	630

3.2 Recipes for National Positioning

For a nation like Nigeria to attain the desirable level of patronage from outsourcing services, a number of issues must engage our attention. Some of these are:

1. **Communication and Internet Infrastructure :** We are in the age of knowledge driven economy, information technology being the key to accelerated development. Therefore, there must be a conscious systematic and strategic approach for putting in place information technology infrastructures like Internet connectivity, and telecommunication at all levels just as emphasis is being placed on good roads, electricity, housing etc.

2. **Knowledge**: This is the key to development in this age, which explains why none of the kingpins of OPEC (Organization of Petroleum Exporting Countries) can be reckoned as among the technologically developed. There is a need to build knowledge both at the vertical and horizontal levels. The vertical level represents high professional competence. The high level education should be research oriented and designed in a way that makes the results practical and applicable in driving forward the economy. For a nation like Nigeria to excel in IT outsourcing there is a need to consciously build core competencies, train and adequately equip our professionals, and students at the tertiary levels with the skills and modern trends of computing. This is especially desirable in fields of software



engineering like software reuse, object-orientation, componentization etc., which bear direct relevance to IT development outsourcing. Sound knowledge acquisition is a major milestone that have been achieved in countries like India and South East Asia, that is presently yielding a lot of dividends. Building knowledge at the horizontal level entails expanding the broad general knowledge of the ordinary citizens, which involves giving broad-based IT education at the primary and secondary levels. This means that the focus of our education should be targeted towards developing high quality research capacity at the higher level and broad-based IT-spiced education at the lower levels.

3. **National ICT Policy**: We have been so privileged that the Nigeria government has seen the wisdom in having a national ICT policy, but we are yet to see the commitment of our national government to implementing polices that are in tandem with the vision of the ICT policy. For example following the Indian model, nothing stops us from having our own form of 'Bangalore' in Nigeria, an outsourcing services friendly zone which will be adequately equipped and made conducive for attracting foreign outsourcing services.

4. **Globalization**: As a nation, there is a need to embrace more of the concept of globalization, opening up for more cultural and economic integration with the outside world. Nigeria, particularly needs to improve on her global image with such initiatives like the Nigeria Image Project [16] which will creates an opportunity to present the good versions of Nigeria to the outside world in order to correct the problem of improper perception that is still being suffered in some quarters, which can have adverse effect on our ability to attract outsourcing services.

3.3 Recipes for Corporate Positioning

There are a number of things that a foreign company prospecting for an offshore outsource service provider looks out for. Some of these include the business profile of the outsource service provider OSP, which is expected to be made available on the web. Some of the other details mostly sought are:

1. The kind of expertise that the OSP has;
2. The size of the OSP in terms of how big it is;
3. The financial stability of OSP and a projection of how long it can continue to do business actively, since nobody will want to do business with an OSP that is about to fold up;
4. The qualifications of the people that will be working on the project, including a scrutiny of presented resumes;
5. Also, evidences of core competencies of the OSP in the particular project area, which should exist in form of project case study reports, company technical reports, research papers etc. [4]

In the light of these, it becomes essential for any organization with a vision of being an outsource service provider (OSP) to be adequately prepared and be correctly positioned.

From the foregoing, a number of things are essential as minimum standard for a prospective OSP. Some of these are:

a. **Presence on the Internet**: Every prospective OSP needs to have a proactive presence on the Internet with a rich, dynamic and highly interactive website.

b. **Track Record of Ethics and Integrity**: Every prospective OSP must be ready to do business the professional way (not the Nigerian way), with a display of high degree of moral ethics in business dealings. Foreign companies tend to be strict on keeping to contractual schedules and deadlines and defaults may most likely be considered as contractual violations [6].

c. **Development of Core competencies**: Such organizations needs to develop special skills and core competencies and certifications in a way that make them fit to compete favourably with what obtains in the developed countries, since this is the only way to attract foreign patronage. Organization should also strive to attain international technical certification such as SEI-CMM Level 3,4 or 5 and ISO-9001 [7]. One unfortunate scenario that must first be reversed is the penchant of big Nigerian companies to patronize foreign IT firms, which means that even software jobs in Nigeria are getting *Bangalored*.

d. **Improved Collaboration**: Within our national frontiers, IT outsourcing or subcontracting should be further encouraged. The bigger companies who have the weight and influence to bid for the big jobs should begin to consider appointing domestic outsource service providers. This will enable more organization to learn the rules and practice of outsourcing and thereby getting

positioned for opportunities to provide offshore outsourcing services.

4. Conculsion

Outsourcing as at today, is a global trend that presents exciting opportunities for developing countries of the world and Nigeria and Nigerian software developers also have the potentials to partake of this, if concerted efforts are made to be correctly positioned as outlined in this paper.

References

Brooks, Jr. F.P. (1986): No Silver Bullet-Essence and Accidents of Software Engineering, www.computer.org/computer/homepage/misc/brooks, 03/03/05

Crnkovic, I. And Larsson, M. (2002): Challenges of Component-based development; Journal of Systems and Software; 61(3), pp 201-212

Crnkovic, I. (2001): Component-based Software Engineering-New Challenges in Software Development; Software Focus; 2(14), pp 127-133

Tesler, B. (2003): "Choosing Your Outsource Service Provider"; http//www.webspacestation.com/it-outsourcing-news/articles, 02/02/05

Tesler, B. (2003): "IT Outsourcing: Advantages and Disadvantages"; http//www.webspacestation.com/it-outsourcing-news/articles, 02/02/05.

Tesler, B., (2003): "Preparing a contract with an Outsource Service Provider"; http//www.webspacestation.com/it-outsourcing-news/articles, 02/02/05.

Tesler, B. (2003): "IT Outsourcing Frequently Asked Questions about Offshore Outsourcing, offshore software development, and a lot more"; http//www.webspacestation.com/it-outsourcing-news/articles, 02/02/05

Kontsevoi, B. (2003): Outsourcing offshore: a forced Trend?; http//www.webspacestation.com/it-outsourcing-news/articles, 03/02/05

Nelson A. (2003): "Outsourcing and Offshore Coders: Good or Evil"; http//www.webspacestation.com/it-outsourcing-news/articles, 03/02/05

http//www.webspacestation.com/it-outsourcing-news/articles/outsourcing-opportunities/

http://www.neoncarrot.co.uk : "Export of good and service of India's IT

Sommerville, I. (1996): Software Engineering; 2nd Edition, Massachusetts, Addison-Wesley.

Szyperski, C. (1998), Component Software Beyond Object-Oriented Programming; 2nd Edition, Massachusetts, Addison-Wesley.

http://www.xprogramming.com.

http://www.bamboo.com.

http:///www.nigeriafirst.org.

An Appraisal of the Problems Areas in the Partnership between Industry and Academia in Nigeria

Uwadia, Charles. O; Longe, Olumide B; Ogege, Francis. I.

ABSTRACT

The benefits accruable to institutions, the industry and the society at large from the collaboration between the academia and the industry in science, technology and national development cannot be underestimated. Unfortunately, such co-operation seems not to be so widespread in our nation for now. What could be responsible for this trend of events? This paper identifies among others corruption, diversion of equipments for private use, managerial bottlenecks, lack of commitments, frequent change of leadership and lack of exchange programs as some of the factors responsible for this scenario. We offer possible solutions that will enhance partnership and produce dividends that will not only be mutually benefiting to the co-operating parties, but will also create an enabling environment for the continuity of such partnership.

1.0 Introduction

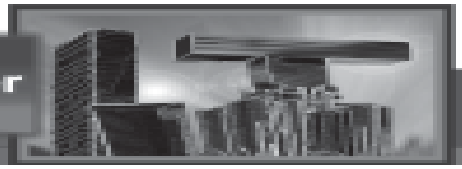
The sharp negative differences between employer's expectation and the actual "on the job" performance of graduates from Nigerian higher institutions has come to the fore especially with the pace at which information technology applications and concepts are rolled out at short intervals. Staff retraining was one of the measures adopted by organizations in times past to make up for any inadequacy among graduate employees. Universities, polytechnics and other tertiary institutions are expected to (at least) be able to lay a proper foundation on which organizations can build on. The problem has, however, become acute, partly due to the "computerization" taking place in all lines of career pursuits and for the fact that an average graduate today lacks some basic background on which industry can infuse current knowledge.

According to Dorris (2003), technological and economic progress demands that employees or job searchers be computer literate. Software from Microsoft, Corel, Oracle among others, are used daily in routine work at all levels of business organizations from non-management to corporate executive positions. In making the transition from manual to automated functions, e-mail and computer automated voice mail often replace face-to-face meetings and live conversation in industries and other organizations.

1.1 The Two Sides of the Coin

Venkat (2004) opined that industry view academics as idealists detached from reality. But simply blaming academia for industry's current state doesn't ameliorate the problems. A situation where the two parties continue to blame each other for the woes currently being experienced will not bring any meaningful solution. As the proverb goes, "it is the grass that suffers where two elephants fight"; the grass in this case being graduates from the various institutions and inefficiencies in job delivery in the industries. It seems obvious that the relationship between the industry and the academia will have to graduate to a symbiotic one for the benefits accruable from both ends to begin to manifest. Academia and industry should collaborate more meaningfully and productively. Although this statement has been made by many, time and again, collaboration has yet to go beyond obtaining grant money and establishing inactive industry advisory committees (Hedström, 1998).

Fig. 1.0" Collaboration in Progress



2.0 Disharmony and Its Effects on the Players

Generally, academics don't value industry experience. Worse, industry simply ignores academic experience. This hostile attitude does not help any professional advancement. It prevents the academic system from attracting seasoned industry professionals who have gained their practical insight by working on the field. In the same vein, some institutions in Nigeria do not count working experience as part of the criteria for placing applicants into academic positions.

Thus, a retired permanent secretary in the civil service, with a Master's or PhD degree could find himself as an Assistant Lecturer or a Lecturer II in a Nigerian University peradventure he applied for a lecturing job. This is contrary to the situation in the advanced countries where so much value is placed on the wealth of experience from people within the industry willing to shift into academic careers.

Similarly, few opportunities exist for academics to spend their sabbatical working in the industry to solve real problems. Another major problem faced by the Nigerian higher education sector is keeping instructors abreast of current trends so as to students how to use the latest IT applications as it applies to various fields. Even if instructors do find ways to keep current personally and then attempt to modify course content to include the use of new technology, they often find it necessary to develop their own teaching materials because there are no up-to-date instructional materials. For example, part of the manpower in software engineering programs should possess substantive practical experience and insight, traits somewhat more valued in engineering disciplines (Heiat and Spiver, 1995).

Employers commonly expect the engineering faculty to possess substantial industry experience because designing and developing a real-world system pose much greater challenges than writing a research paper. In contrast, academia typically favours publications over practical experience. Academia alone cannot extricate itself from the current situation without the active participation of industry and professional organizations such as the Nigerian Computer Society (NCS), Nigerian Society of Engineers (NSE), etc. Academicians dominate professional societies, thus these societies primarily reflect academics' needs such as organizing conferences and publishing magazines, journals, and conference proceedings.

2.1 Other Problem Areas.

Apart from all that has been said so far, there are other problem areas that we have to identify and tackle before we can begin to see the gains from the collaboration we all are yearning for. These are summarised as follows:

(a) Embezzlement of Provided Funds and Grants: The problem of embezzlement of funds provided by the industry and organisations for the purpose of developing projects that will be of great benefits to the industry is no longer news. One advice that the authors' feels will work at this time is for endowing firms to release funds as progress is made with the projects under funding. This will enable industry monitor the progress of the projects for which funds are released and minimize corrupt practices.

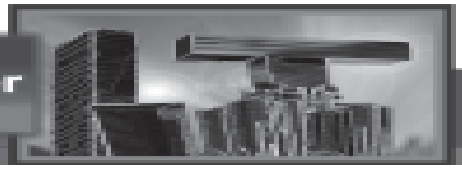
(b) Lack of continuity in leadership: Different academic leadership at departmental, faculty and management levels pursues different goals. This lack of continuity does not augur well for collaboration between academia and industry.

(c) Diversion of funds and equipments sent in by the industry for private use among academics is a common occurrence in our institutions. This breeds lack of trust and discourage inputs from the industry.

(d) Donation of obsolete equipments from the industry to Nigerian institutions (mostly IT equipments) does not represent genuine interest in the academia on the part of the industry. Such equipments rather than enhancing the knowledge of the students and staff for industrial applications, retards and takes them back in time. More money is also spent trying to maintain them.

(e) The provision of an enabling environment by the receiving institution for persons from the industry to come in on leave to impart knowledge on the students and staff is not a regular occurrence in our institutions. This exchange when operational will ensure a free flow of knowledge and understanding that will be mutually benefiting to both parties. Industries will also have to open up their doors for academia to come in on sabbatical in order to understands current industrial processes so students can be tutored along this lines.

(f) Using students in the industry as casual labourers and making them work in more dangerous aspects of industrial processes would not augur well for the development of the academia. This is an obvious abuse of the



student's industrial experience schemes. Industries and organisations must ensure that students placed on industrial training are fitted into appropriate positions within organisations during industrial attachments (Lum, 1994)

(g) The quest for immediate gains and lack of vision for long-term benefits on the part of the industry will not propel them to collaborate with the academia.

(h) When students on IT are cheated and not paid commensurate wages, students will be discouraged from pursuing training in such establishments talk less of pursuing a working career.

(i) More forums must be established that will bring together the academia and industry for the purpose of interacting to the benefits of both parties. Mission, vision, goals, and objectives of industry must all be closely examined and aligned with reality, while balancing tactical needs with long-term strategy. Some energy should be channeled away from introducing more conferences and publications that only results into the explosion of more or less the same information packaged differently and appearing in a plethora of forums (Macmillan and Hamilton, 2000).

(j) Certification and licensing: Professional societies should champion the need for and administration of certification and licensing. They should also liaise with government, business, and industry to educate these sectors about the significance of hiring licensed professionals and the process involved in that licensing. Currently, the market treats the IT workforce as disposable labour. Societies should actively seek to improve the stature and public image of Nigerian professionals and address issues that affect the careers of graduates and practicing professionals.

3.0 Intellectual Property Rights and Collaboration

Some form of collaboration requires contractual agreements between industry and academia. An example is the implementation of a prototype developed by the academia into real life project. Probably the most important contractual conditions are the ones regarding Intellectual Property Rights (IPR), confidentiality, and the financial obligations.

Academic administrators seem to attach more and more importance to protection of IPR. Consequently, one should reserve sufficient time for negotiating IPR arrangements during contract negotiations. Intellectual Property Rights define and protect the ownership of information. IPR include patents, copyrights, utility models, design patents, design rights (both registered and unregistered), as well as other forms of protection offered by law to inventions, designs, technical information and applications (Longe, 2004).

3.1 Possible Information for Protection

Information requiring IPR protection can include drawings, specifications, photographs, samples, models processes, procedures, instructions, software, reports, papers, correspondence and any other technical or commercial information such as data and documents of any kind. As a result of legal considerations this definition of information concentrates on items, which are made available in a hard-copy form. Therefore, if any information needs to be protected which was given during meetings in a soft-copy form (e.g. shown on screens, or explained orally), then this information should be minuted in hard-copy form immediately after the meeting.

Regarding IPR one should distinguish between foreground information and background information. Foreground information is generated during the execution of the collaboration project. Background information is pre-existing information which is needed in order to use the foreground information. It is important to agree on the way in which the various co-operating partners have access to each other's foreground and background IP. Industrial partners will generally want to have a comprehensive license that gives them all rights (at least for their field of applications) to commercialize the "inventions" done during the co-operation project (i.e. the foreground information).

Academic partners will generally want to have financial (and other) compensations for transferring the rights on their part of the foreground information to the industrial partner. This compensation may come in the form of a one-time lump sum, or a fee for every time a product is sold that uses the invention. Another form of compensation may be in a sustained relationship: a mid-term or long-term funding of the academic group by the industrial partner. This should be arranged in the contract.

An error often made by academic partners is that they may estimate the value of specific inventions unaffordably high. As on any market, excessively priced goods which are not essential may not be purchased,

and the potential buyer may seek a lower cost replacement of comparable utility.

3.2 Publications Versus confidentiality

Academic researchers need to be able to frequently publish about the results of their work. Although industrial partners will recognize this need, their interests should be protected, e.g. by applying for patents before publication, and/or by omitting some of the finer implementation details from the publication (which is often possible without devaluing the publication). Industrial partners also need to ensure confidentiality of the internal information that was disclosed to the academic partner. In a good cooperation both partners should find a mutually agreeable way to act in their own interests while not hurting the interests of the other party.

4.0 Conclusion

This paper has been able to elucidate some of the problem areas in the collaboration between academia and industry in Nigeria. We have also offered possible solutions that will enhance partnership and produce dividends that will not only be mutually benefiting to the co-operating parties, but will also create an enabling environment for the continuity of such partnerships.

In addition to all that has been said so far, working through professional societies to shape the curriculum, providing real-world case studies for classroom use, establishing programs through which institutions staff you can spend sabbatical time in industry, and exploring other avenues to strengthen cooperation would benefit industry. Educators must recognize the profession's current reality , forge a vision for improving it, devise a strategy for realizing this vision, then provide sound leadership for its implementation. If we do not, "survival of the fittest" alone will determine the profession's Darwinian future.

References

Doris, D. (2003): Partnerships Between Industry and Academia In Information Technology. Journal of Education for Business. Vol. 1, No. 3

Venkat N. G (2004): The Computing Profession at a Crossroads. IEEE Computer Society Journal. Available at <http://www.computer.org/computer/archive.htm>.

Heiat, H. and Spicer, G. (1995): Future Software Training Needs: Contrast in Needs as Perceived by Business and Academia. Journal of Information Systems Education.

Hedström, K. (1998): A theoretical framework for analysing co-operation between practioners and researchers. IRIS 21, Saeby, Denmark, University of Aalborg.

Longe O.B (2004): Software protection and Copyright issues in Contemporary Information Technology. Unpublished M.Tech Degree Thesis. Federal University of Technology, Akure, Nigeria.

Lum, L. (1994): Firms Hiring 4 Out of 5 of Their Interns, Inland. New York Times Service; Valley Daily Bulletin, January 30.

Macmillan, G. S. and Hamilton, R.D (2000): Corporations Need to Publish - or Perish. Research & Technology Management. Vol. 43 No. 1

IT Curriculum and Employment Situation in Nigeria

Z. Lipcsey, E. E. Williams, E. O. Ukem, R. C. Okoro

ABSTRACT

Information technology (IT), as an innovation for handling information, is concerned with the various means of generating, processing and transferring information using computers and telecommunications gadgets. Education basically is aimed at equipping the citizenry for future employment. Therefore for education to be effective it needs to be properly delivered. This means that the curricula used in this teaching must be such that the candidates at the end will fit into the required labour positions in a given country. The curricula, which handle the IT training in Nigerian Universities and also worldwide are the curricula of various computer science programmes and computer engineering programmes with possible specializations (using University of Calabar as a case study). The aim of this paper, therefore, is to analyze the curricula of these programmes and find out, how relevant they are to the present IT expectations and to the fast changing employment situation in Nigeria. This work examines the curricula of these programmes for IT education in Nigeria with view to finding their relevance to employment opportunities in the country and abroad. The job opportunities of graduates in IT field vis-à-vis other fields are also discussed.

INTRODUCTION

The notion of curriculum is not new - but the way we understand and theorize it has altered over the years. There remains considerable dispute as to its meaning. It has its origins in the running/chariot tracks of Greece (ref). It was, literally, a course. In Latin curriculum was a racing chariot; *currere* was to run. A useful starting point for us here might be the definition offered by John Kerr in his standard work on the subject. Kerr defines curriculum as, 'All the learning which is planned and guided by the school, whether it is carried on in groups or individually, inside or outside the school (Kerr, 1999).

Generally a curriculum is setup with a particular purpose in mind. The process of forming the curriculum is divided into five categories (Wheeler, 1979) namely:

- The aims, goals and objectives
- Selection of learning experience
- Selection of content
- Integration of learning experience and content
- Evaluation.

One of the major difficulties of curriculum process is the transition from general aims to the particular objective of the classroom. To achieve the required aim there is need to state the ultimate goals, derive the mediate goals and finally from these setup the proximate goals with a view to planning the specific objective. The ultimate goals are the expected outcome expressed as patterns or categories of behavior. IT is not left out in this curriculum process in that once the five phases of curriculum development is followed, the ultimate goal will be such that will benefit the graduates of such curriculum. The main problem today of the IT curriculum is that the processes of curriculum formation are not taken into consideration. In a situation where an expected IT graduate is mostly saddled with the philosophy of computing instead of the theories and practice of computing, will not fashion the graduate to compete with his/her counterparts from other parts the world over.

Currently Nigeria is on the receiving side of the outsourcing process while the overseas countries are the sources. With the world now effectively a global village, events and trends elsewhere do have effect here at home, which have to be carefully analysed, the position fully understood and correct conclusion can be derived. The IT curriculum needs to be updated and new development will have to be carefully incorporated. Considering that a curriculum is supposed to prepare somebody to operate for life while certification examinations target short-term skill for fast productivity, professional examinations should be enforced while certification examinations should be encouraged. IT practitioners must be equipped to consolidate the IT industry locally (common with overseas target), while at the same time being able to tap into the global international trends.

IT CURRICULUM PROCESS

It is helpful to consider the ways of approaching curriculum theory and practice in the light of Aristotle's influential categorization of knowledge <http://www.infed.org/biblio/knowledge.htm>(http://www.infed.org) into three disciplines: the theoretical <http://www.infed.org/biblio/knowledge.htm>, the practical and the productive.

Here we can see some clear links - the body of knowledge to be transmitted, which is classically valued as 'the canon'; the process and praxis models come close to practical deliberation; and the technical concerns of the outcome or product model, mirror elements of Aristotle's characterization of the productive.

THE NEED FOR IT CURRICULUM IN NIGERIA

The curricula, which handle the IT training in Nigerian Universities and also worldwide are the curricula of various computer science programmes and computer engineering programmes with possible specializations. In other words, we do not have any standard curriculum to follow and plan for the future of IT operators, including the knowledge package for them. If we were to have, such a standard IT curriculum should:

- Produce a model to follow in our universities, setting up the aims and objectives, and strategies to achieve such aims and objectives.
- Plan and implement certification examination at various levels.
- Monitor future trends in IT profession and also make changes in curriculum to reflect expected future needs and employment.

Setting a model to follow: Setting up a model to follow in training clearly defines the aims and the objectives, and the goals, and also outlines strategies to achieve the stated goals. Considering Aristotle's model discussed above, we see that the goal is the product. The type of product is planned for during the initial stages of development.

Plan and implement qualifying examinations: The curriculum should allow for phased progress within the profession. A similar strategy exists in some other professions, for example in accounting and engineering. Accounting professionals take qualifying examinations

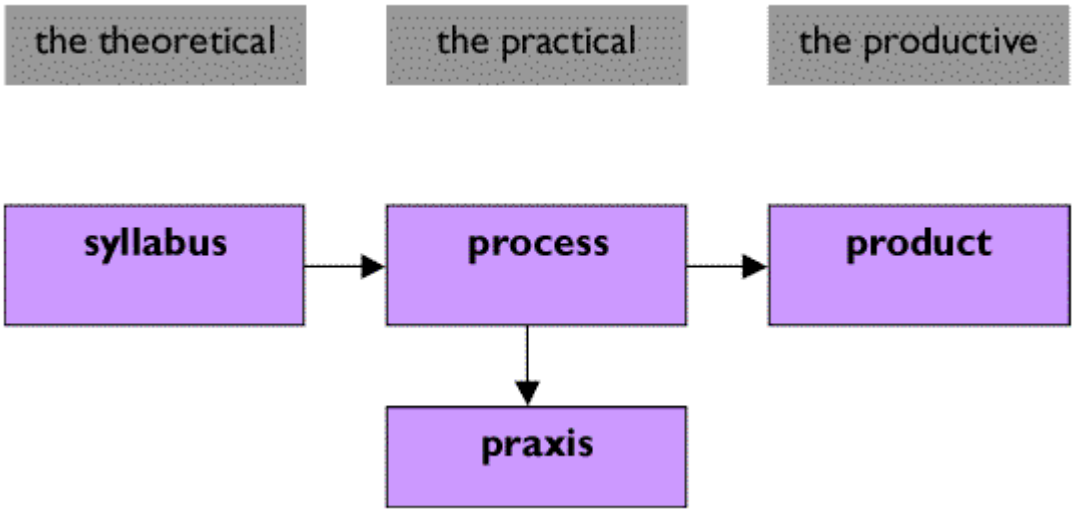


Fig 1. Curriculum framework (Source: Aristotle: http://www.infed.org)

at various stages, and perform certain job functions, which are related to, and limited by the stage reached. In engineering, the professionals are categorized according to the qualifying examinations taken. Thus, there are engineering technicians, technologists, and engineers. Similarly the IT curriculum should allow for such categorization. Certainly, IT professionals are not all required at the same level. The work environment demands that there be such categorization.

Monitor trends in IT profession: The IT curriculum should be dynamic in nature. Trainers should monitor and try to forecast the future trends and also make changes in the curriculum to reflect expected future employment needs. For example, a research carried out by Paula Rice shows that over the next five years, IT professionals will also need to continually upgrade their technical skills to remain competitive (Rice, 2004). But, more importantly, they will need to remain flexible and adaptable. For example, a 'well-rounded ' skill set is important. Large companies like IBM or Hewlett-Packard can afford to hire three separate people to handle systems administration, back-up and recovery, and e-mail administration. But small to medium sized companies must hire one person to handle all three responsibilities. An IT curriculum should monitor the needs of the industry and produce this type of multitasking professionals, where knowledge in single areas would not be as deep as it was in the old days when five people worked together to do one job.

Apart from training multi-talented IT professionals, there is going to be an increasing integration of work and personal time in the near future. This is the result of availability of powerful and portable computers. A standard IT curriculum should be able to fashion our future IT professionals towards achieving this type of work life.

Companies and other business organizations aim at maximizing profit. As Joseph Williams puts it, IT education becomes more specialized in the future; and that 'the idea of specialization is that nations will invest in what provides them natural competitive advantage in the global economy' (Williams, 2004). Currently more US companies are outsourcing jobs, and countries such as India and China take advantage of their low-waged but disciplined and relatively well educated citizenry to provide robust infrastructure for call centres, support centres and software factory type of jobs. Williams' work shows that ironically, the demand for off-shore IT workers is starting to create pressures on wages; consequently, several national governments such as India and China are investing heavily in education to increase the pool of IT workers and keep their competitive wage advantages. Nigeria can work also towards contributing to this pool of IT workers in the world. Nigerian IT professionals need to be able to sustain the growth of the industry at home while at the same time not losing sight of global trends, but be in position to ' flow along'. The time to start is now!! What can attract any company to outsource in Nigeria apart from wages and training specialization is our ethics. We need to reconsider our work ethics and change our work-life if anything meaningful can happen.

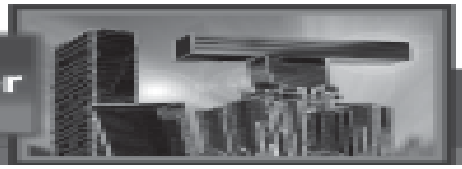
EFFECTS OF NON-STANDARD CURRICULUM

From the fore going it is obvious that curriculum formation is useful whenever the basic factors are considered as being important. When this happens there is a proximate value that will help in the development of not only the individual but also the society. Most youngsters today engage in roadside computer training. One wonders what the curriculum used in such places is like. Is it one drawn up by knowledgeable persons or one by the sole proprietor of the training school? The obvious simple answer is that it is the one by the sole proprietor of the school. If that is the case, could one honestly consider the outcome of this as a good IT training? If the answer is no, then something has to be done to make sure that an appreciable standard is put in place to guard against this ugly situation. The effect of this type of training is not only that the candidate cannot fit into the society but also that the profession as a whole is being dragged into the mud. In our opinion, the way forward is not to discard or outlaw these schools. Obviously a need exists for the caliber of professionals they are attempting to produce, as evidenced by the known demand for them. They should be serving a useful purpose, only that the quality is doubtful. Measures should be taken to ensure that they apply a curriculum properly designed for that level of manpower. We feel it is not beyond the Nigeria Computer Society (NCS) and the Computer Professionals Registration Council of Nigeria (CPN) to bring this about.

Today every establishment wants IT professionals. This could only be achieved if the curriculum used could meet the required standard, which will in turn give the candidate the expected confidence to perform. Since this technology seems to be what everybody wants, there must be standard to avoid "quarks" in the profession.

CONCLUSION

The IT curriculum in Nigeria must be overhauled with view to having more reasonable structure that will meet the International standard. This will not only give the



graduates the confidence to work but will also offer them the opportunity to work anywhere in the world. To achieve this, knowledgeable professionals should be involved in IT curriculum development.

Currently Nigeria is on the receiving side of the outsourcing process while the overseas countries are the sources. With the world now effectively a global village, events and trends elsewhere do have effect here at home, which have to be carefully analysed, the position fully understood and correct conclusion can be derived. The IT curriculum needs to be updated and new developments need to be carefully incorporated. IT practitioners must be equipped to consolidate the IT industry locally (common with overseas target), while at the same time being able to tap into the global international trends.

The general unemployment situation in the country at the moment is rather high, compared with that of the industrialized countries. A properly developed IT curriculum would prepare more Nigerians to take up jobs locally and also anywhere in the world, thereby helping to reduce overall unemployment figures.

REFERENCES

John Kerr 1999 curriculum theory and practice <http://www.infed.org/archives>

Joseph Williams (2004): IT Education: More Specialized in 2010, IT Professional. Technology Solutions For The Enterprise. IEEE

Paula Rice (2004): Maintaining Work-Life Balance in 2010, IT Professional. Technology Solutions For The Enterprise. IEEE

Wheeler D.K. (1979) Curriculum Process.

<http://www.infed.org>

<http://ade.state.az.us/standards>

Enhancing Software Engineering Practice through Result-Oriented Academia/ Industry Collaboration

Adewumi, A.O; Sawyerr, B.A; J.O.A. Ayeni

ABSTRACT

There have been some informal ways by which the Universities and industrial organizations work together including student industrial attachment, job placement and industry projects aimed at completing a degree program. However, giving the peculiar circumstances and situations in developing countries like Nigeria, such collaboration has not been yielding expected results. Meanwhile, many developed countries had made great impact in software engineering practices and they had attracted a lot of governmental interest due to the formal collaboration that exist between the academia and the industry. What then makes collaboration successful? In this paper, we review some of the factors that had helped developed countries achieve success. We then discuss and advocate for a more formal collaboration between universities and the industry organizations in order to enhance software engineering skills for practitioners and the marketability of IT achievements and advancement generally. We provide insights into what should be the nature, benefits, and process of such formal collaboration in Nigeria.

1.0 Introduction

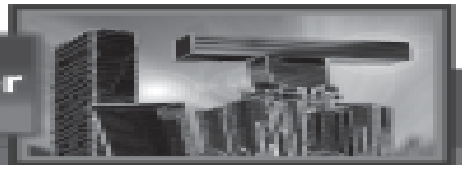
This is the strategic time to start investing in industry-university partnership that put the product of university research work in the economy - Richard Atkinson, President, University of California

Collaboration is defined as *a formal, joint effort by a university (or universities) and a business or government organization(s), where each party provides specified products and services to achieve common goals* [6]. This was the definition given by Dennis Frailey during his talk which eventually motivated the Industry/ University (I/U) subgroup of the Working Group on Software Engineering Education and Training (WGSEET) to begin investigating reeducation collaborations between academic institutions and industry for non-software engineers.

Partnerships and collaborations between educational institutions and industry, for the purpose of research and education are an important means of establishing and maintaining relevance to practice in the engineering curriculum [15]. They facilitate sustained contact between faculty, students, and practicing engineers. They also provide a means for corporate organizations and industries to access expertise and facilities at university campuses, and gain visibility among potential employees.

As with all successful relationships, collaborations must be based firmly on mutual trust, respect, and a willingness to understand and respect the institutional imperatives of the other partners [15]. Collaborations involving industry and universities can be beneficial for all participants. In order for these activities to achieve their potential, however, a solid foundation must be established. This foundation consists of a clear common understanding of each partner's long term vision for the collaboration (and compatibility of those visions), and a plan that involves sufficient resources to move the effort through its initial stages. However, motivating creative and smart people to work together can be one of the hardest aspects of industry/academia collaboration, especially where various arrangements and agreements need to take the commercial interests of industry into account as well as the highly aspirational interests of individual researchers [9].

In most developing countries like ours, universities and industry organizations have traditionally maintained *informal* ways of working together, including student industrial training programme (SIWES), graduate employments, and industry capstone projects to complete a degree program. However, a *formal* collaboration between a university (or group of universities) and an industry organization (or group of organizations) was proposed in [13] to meet the critical needs of software engineering education and training. A number of questions were raised in view of having successful industry/university collaborations for software



engineering education and training. These are:

- Why collaborate? What are the benefits?
- How would we start, and then operate, collaboration?
- What makes collaboration successful?

These questions formed the basis of a survey carried out by the SEI (Software Engineering Institute) Working Group on Software Engineering Education and Training. The survey was conducted for both academic and industry representatives of the 23 collaborations documented in the 1997 Directory [1]. After thorough examination and analysis of the research survey, the SEI working group came out with a number of facts as well as a formal model for establishing successful industry/academia collaboration. We present some of the results of the survey in this paper.

2.0 The Need for Collaboration

The main reason why we need collaboration between the academia and industry is the current trend of poor funding of tertiary institutions in Nigeria especially the nation's universities. This problem could be seen as a great challenge by the academia and industry to seek for more result-oriented and successful collaborative efforts that could sustain our systems. This collaboration was actually one of the secret of the development of universities in the developed world. A research study carried out by Kim, S. (1997) reported the multiplication of university-industry collaborations United States in the past decade due to response to the competitive pressures for U.S. industries in the global marketplace and the cutback of the federal R&D budget owing to rising federal deficits [10]. As the funding by the U.S. government declined academics were forced to seek funding from industry. Many academics find industrial funding attractive as this involves less control and less red tape than federal funding. Moreover, the time to produce research proposals will be considerable reduced as governmental regulations related to scientific and financial accountability is excluded. Academic researcher can then achieve greater efficiency and more independence in their work.

As evident in our universities and the country in general, universities and even industry now exist in a harsh and competitive economic climate. Cost of research has escalated. Davies R.M (1996) supported the need for universities to seek financial support from industry as conventional sources of funding via viable collaborative research programmes [2].

In [13], the result of a survey of formal industry/ university collaborations conducted by the SEI Working Group on Software Engineering Education and Training in 1997-98 was presented. They identified the purpose of collaborations as meeting the software engineering education and training needs of adult learners through joint ventures such as graduate programs (degree and certificate) and professional development activities (customized classes, seminars, forums, and conferences).

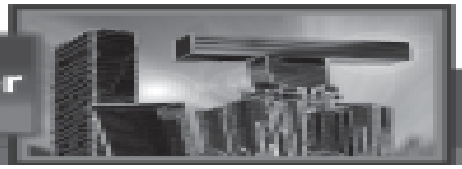
There are many reasons for forming industry/university collaboration. According to SEI working group on software engineering education and training [1,8], these reasons include fulfilling an organization's education mission, accessing education and training resources, gaining competitive advantage, addressing business growth, achieving cost savings, enhancing organizational reputation, increasing revenue, accessing research and tool resources, and providing a staffing source.

2.1 Cases of Collaboration

The prevalence and vitality of research partnerships between industrial organizations and universities have increased dramatically over the last two decades in the developed countries [8]. Collaborative partnerships have become strategic assets for companies that face increasingly rapid technological change, increasingly intense international competition, and diminishing in-house research resources [8].

A number of existing collaboration between academia and industry from Germany is reported in [4]. These include an informal collaboration in form of small workshops organized by professional organizations, meetings, discussions; Small fully funded projects; Funding of PhD-projects; and Cooperative projects with public funding. Graduates were reported as the most important means of transfer from academia to industry (and vice-versa). Some graduates start with the R&D departments or groups of a company involved in the collaboration. Moreover, graduates with industrial experience are in many cases preferred as university lectures.

In 1995, the Working Group on Software Engineering Education and Training (WGSEET) was formed with the mission of improving the state of software engineering education and training [7]. An ad hoc group of international professionals from academia, industry, and government within WGSEET was formed with a focus on the education and professional development of software practitioners through degree programs, continuing education, on-the-job training, etc. The Industry/University (I/U) subgroup is a subset of WGSEET members whose focus is to explore and foster



collaborations between academic institutions and industry. The Industry/University (I/U) subgroup investigated active collaborations between companies and universities in which employees without formal software education are reeducated to become software engineers [6].

The case of Norsk Hydro, a major worldwide producer of light metals aluminum and magnesium and a leading producer of aluminum alloy extrusions collaborating with the Norwegian University of Science and Technology NTNU, which conducts research on all aspects of aluminum production (smelting, casting, extrusion) as well as well as application of extrusions in marine, automotive, and civil construction was reported in [15]. The collaboration started with series of planned meetings between the company and the Departments involved where a number of issues were cleared including, general requirements and expectations for sponsored research, intellectual property agreements, and possible modes of interaction (joint research, student and faculty exchanges, industrial internships and employment).

One of the agreed areas of collaboration was sponsoring of some relevant research projects on which graduate students were admitted for their Masters and PhD programmes. Chairs were also funded while a number of students were granted internship and/or employment. Workshops were also jointly organized and sponsored to discuss and share research ideas that could benefit both parties. Perhaps most importantly, the partnership provides an important element of industrial relevance and cultural diversity to the education of students in the participating programs.

Another case study that is instructive is the collaboration between the Cardiac Rhythm Management (CRM) organization at Guidant Corporation and Embry-Riddle Aeronautical University (ERAU) [11]. The aim of the collaboration is on technology transition. This involves developing the professional skills of the software engineering students by providing them with real world problems, scenarios, and modern techniques in a student-oriented research laboratory. The gains for the industry includes but not limited to skilled and market-ready graduates, and tools for architecting and designing software for critical-safety systems.

Two series of conference was held in Czech Republic in August 1999 on the 'Progress through Partnership' at the Technical University of Ostrava and the Technical University of Prague, Czech Republic respectively [14]. The purpose was to bring a team of U.S. Engineering educators for participation in the conference (ICEE-99) so as to foster the formation of collaborations that would promote international activities in curricular development and increase international awareness among their students. According to the report submitted for a post-conference workshop [14], the following were agreed upon:

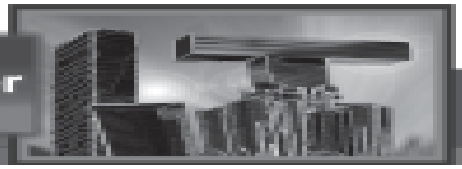
1. That International partnerships and alliances in education and research are the key factors for progress through partnerships.
2. Faculty/student exchange program is considered as an ideal vehicle to put international university programs into action.
3. Global industry-university interactions are instrumental in improving education, training, and cultural understandings of the future engineers for career opportunities in global markets.
4. In order to provide a forum for international collaborations and alliances, the formation of an international society for engineering education or a network for engineering education and research was deemed necessary.

As the number and durability of joint ventures rises, understanding of the factors that contribute to their success and to the satisfaction of the partners grows [8]. Many assume that industry looks to academia primarily as fountain of basic, leading-edge research. Entrepreneurial academics turn to industry for additional funding and for access to state-of-the-art facilities and equipment.

One major observation from past research findings on collaboration is the recognition that immediate commercial return is neither the only, nor necessarily the predominant, motivation for companies to enter into research relationships with university scientists. Rather, many industry leaders state they value universities most as wellsprings of scientifically trained personnel (Government-University-Industry Research Roundtable, 1997) [12] and as windows on the future of technology.

3.0 Models of Collaboration

In their assessment of industry's interest in university research, it was reported in [8] that at least two sets of factors contribute to continuing uncertainty about the means of predicting or evaluating success in industry-university collaborations. First, the range of models or structures for collaboration is extremely diverse. A corporation may fund a specific project or an individual investigator as a consultant; a consortium of companies may contribute to university-affiliated research or technology centers; or corporate, state, and federally-funded academic investigators may collaborate, either on short-term projects or on long-



term programs. As the structures of these partnerships vary, so do the expectations of participants and the criteria by which success can be defined and evaluated. Second, the general use of quantitative indicators of performance and success, such as the number of patents, publications, or new products, fails to capture the diverse benefits from joint research ventures that both corporate and academic participants seek.

A number of models of collaboration between the industry and university have been proposed in literature [13]. A summary of possible model is given below:

- Single Company/Multi-University Model
- Multi-Company/Single University Model
- Single Company/Single University/State Support Model
- Multi-Company/Multi-University Model
- University/consultant collaboration

The choice of model depends on the extent and ability of the interest group to initiate collaboration. The objectives and goals for setting out the collaboration might also be an important factor in the choice of the model.

Meanwhile, the most important thing is not necessary the model but the attempt to knowing what the industry/university collaborations should offer. It is very important to devote sufficient attention (time, energy, patience and alertness) to the phase in which the collaboration is formed, regardless whether one is setting up a 2-party collaboration or a multi-party consortium [5].

According to [8], successful collaboration is meant to offer a variety of software engineering education and training activities including classes, seminars, conferences, workshops, and certificate and degree programs. The title and nature of courses to be covered will be as agreed to by the partner and such as will benefit both or at least fulfil their stated objectives.

3.1 A Formal Model for Industry-University Collaboration

One of the most crucial steps in a successful collaboration is initiation [8]. Most successful collaboration started by way of deliberate seeking out a new contact with industry, university, or government to establish the collaboration, while few just expanded an existing relationship to get started. Faculty Professors having link with industries (e.g as consultants) could also initiate collaboration. Formal agreement such as a charter, memorandum of understanding, or contract could be evolved as the basis of such collaboration.

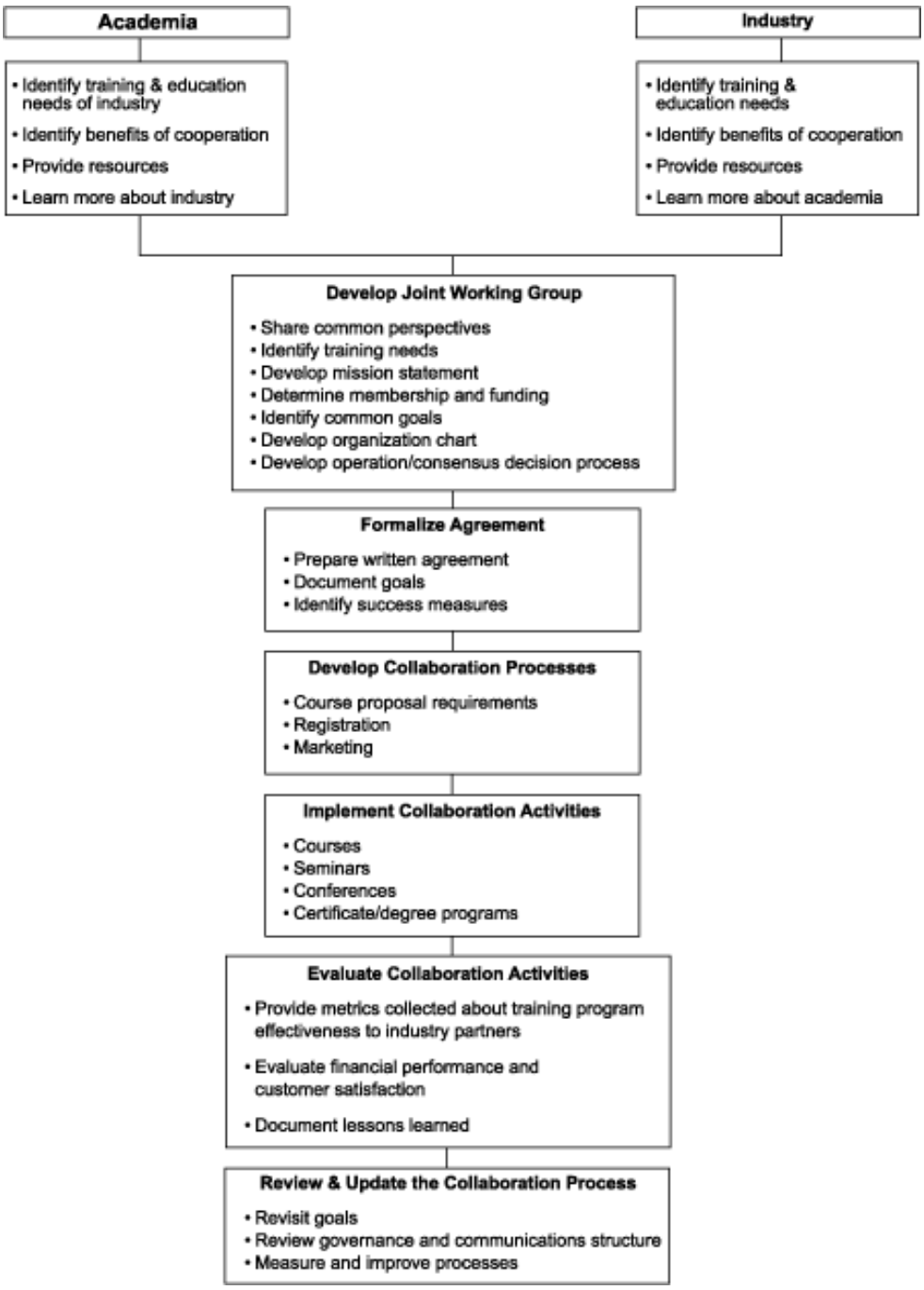
A formal model for initiation and implementation of a successful collaboration as provided in [8] is presented in Figure 1.

3.2 Guidelines for Choosing Collaborative Partners

Gerritsen, F.A (1999) gave some guidelines regarding the choice of collaborative partners as follows:

- *Optimization = seeking a minimum, not a maximum* - in term of mutual advantages of the partnership
- *Quality* - of the potential partner
- *Reliability* - in terms of *integrity* (maintain high ethical standards), *commitment* and *responsibility* (do what was agreed, in spite of problems encountered), *transparency* (be open about what is being done, by whom, when, and how), *confidentiality* (be capable of properly balancing the academic interests of publishing versus industrial interests of patenting or keeping secret), and *loyalty* (give negative feedback only to partner, not to third parties).
- *Complementary* - in terms of each others capabilities (and knowledge, experience, etc.).
- *Expectations* - must be clearly stated, understood and agreed upon.
- *Mutual dependence* - An optimization algorithm would tell you to cut away partners on which you are insufficiently dependent.
- *Chemistry, mutual respect and understanding* - good degree of inter-personal compatibility, mutual respect and understanding is indispensable. Understanding of the limitations imposed by the partner's local environment is also very important.
- *Competition* - It is very hard to successfully co-operate with a direct competitor for a prolonged amount of time.
- *Duration* - Although sometimes this may be hard to work out immediately, it is important to assess what the potential partner's intentions are regarding the intended duration of the co-operation.

Figure 1: Academia and Industry Collaboration Process (Adapted from [8])



4.0 Benefits of successful collaborations

4.1 Possible Conflict of Interests in University-Industry Collaborations

Universities are described as one of the most complex communities that have been devised by humankind [3]. Society has many and high expectations of universities. Universities are full of clever people, tolerant of complexity. The most important *impacts* for the university are the impacts on students and the impacts on faculty [3] as students and faculty are the primary producers of the university's basic outputs: discovery and learning.

A conflict of interest occurs when a reasonable act that would serve one interest at the same time compromises

another interest. For example, for *students*, the principal goal is evidently education at different level (graduate or undergraduate) with different goals (general, disciplinary, skill specific). In the context of industry interactions, another interest is employment interests which include the effectiveness of the training side of education, reputation of the university or program, and informal and formal contacts with employers. To the same extent that students are involved with commercially valuable research, they also have significant financial interests in the intellectual property they create.

For *faculty,* members share many of the student interests, not only the education interests but also, particularly for graduate students, their employment related interests. Some professors are also seen as having interest in community service, most notably for us, in building productive partnerships with industry. Finally, there are very important *self* interests, including prestige, financial reward, and intellectual adventure. However, for the *Industry* the primarily is on how they might provide returns to its shareholders. Thus commercial success is the strongest interests of most industries.

An attempt had been made to describe academic and industrial cultures to clarify the extent of conflict of interests between universities and industry [10]. Academic culture was defined as the collective, mutually shaping patterns of norms, values, practices, beliefs, and assumptions that guide the behavior of individuals and groups in an institute of higher education and provide a frame of reference within which to interpret the meaning of events and actions on and off the campus [10]. Academic culture influences individual faculty members through institutional goals, size, complexity and missions. The definition of academic culture was expanded with three basic academic values introduced [10]: The first basic value is the pursuit and dissemination of knowledge as the purpose of higher education. The second basic value shared by faculty is autonomy in the conduct of academic work. The third shared value is collegiality, and it is demonstrated in a community of scholars that provide mutual support and opportunities for social interaction and in faculty governance. Thus, according to faculty members, an ideal academic community is a college or university in which the pursuit of learning, academic freedom, and collegiality are strongly held values

On the other hand, industry flourishes by controlling knowledge [10]. The industrial imperative is to garner a profit; knowledge with which one can generate a new product or process which is private property for industry. Therefore, industries tend to protect themselves by wielding their proprietary rights over knowledge generated by a university. As industry has grown more inclined to assert proprietary rights over research findings generated by faculty members, the incompatibility of such protectiveness and traditional values of open research has become obvious.

Research publication is a fundamental value in academia. Reputation is dependent on publication in refereed journals. For industry, however, publication may reveal critical information essential for a commercial product. Therefore, funding companies require a delay of publication in order to hold a technological advantage. Research findings may be published when they can no longer help the competition in the marketplace.

One of the fundamental fears expressed in the past is whether university-industry collaborations would threaten traditional academic values [10]. Of particular interest is the extent to which R&D collaborations affect the traditional missions of the university including teaching and research. Although some research reports were said to have held that view earlier, but the important conclusion of an empirical study carried out by Kim, S. (1997) is that academic involvement in industry-related research does not seem to affect the university mission of teaching and research negatively. Better yet, the benefits universities receive from university-industry collaborations appear to enrich professors' educational roles.

4.2 Potential Benefits of a Successful Collaboration

The primary observation of a survey conducted and reported in [6, 7] is that these collaborations provide significant benefits for all three major stakeholders: the industry partner, the academic partner, and the students. Industry feedback indicates that companies benefit from a larger, more highly trained workforce with a higher level of knowledge and skills while retaining large stores of domain knowledge held by the existing employees. Universities benefit from increased visibility of their programs, as well as from the dissemination of a real world vision of software engineering throughout the faculty. Students benefit by becoming more highly skilled workers, with higher job satisfaction and more career mobility [6, 7, 11, 13].

However, as stated earlier, not all benefits from successful collaborations are immediate and tangible. A summary of the benefits (long- and short-term) derived from successful collaboration that exists between participating industries and universities in the survey carried out by the SEI working group on software engineering education and training [1,8] is presented

below:

Financial benefits

- Increased university and partner revenue
- Cost savings
- Reduced training costs per employee
- Some revenue for program development
- Increasing support for research programs

Business growth

- Opportunity for follow on business
- Attracted students for degree programs
- Enhanced partner's marketing
- Extended the reach of the university

Fulfilling an organization's education/training mission

- Top quality training
- Best use of our company's training resources
- Fulfils company training requirements for specific courses
- Occasional use of academic knowledge and courses to supplement industry training

Enhancing organizational reputation

- Better name recognition for both parties
- Public relations benefit and local credibility from the partnership

Providing a staffing source

- Excellent sources of interns and potential hires

Others

- Knowledge of workings of the opposite sector
- Sharing of knowledge and experience of member companies
- Insight into member companies' training programs, issues, problems, and experiments
- A community of practitioners able to share their expertise as a community of learners rather than as competitors

5.0 Industry-University-Government Roundtable for Enhancing Engineering Education [IUGREEE]: A Challenging Example

At this juncture, we would like to present a case study of what was done to foster industry-academia collaboration in the developed world as a way of challenging the three major stakeholders in Nigeria (government, university, industry) to follow suite. We present a case of Industry-University-Government Roundtable for Enhancing Engineering Education [IUGREEE] formed in the United State as reported in [12]. It is our believe that our existing IT and IT-based industry, government parastatals and ministries as well as professional bodies such as NCS, CPN, ITAN, etc. will learn from this case study.

The Industry-University-Government Roundtable for Enhancing Engineering Education [IUGREEE] was formed in 1995 to provide a collaboratively developed voice, vision and *action agenda* for engineering education reform. Most past efforts to reform engineering education have been developed predominantly from a university perspective. A strong, sustained industry voice and vision that clearly articulate *strategic* changes in engineering practice, changes that should prompt education reform, were however deemed necessary hence the formation of the Roundtable. IUGREEE used the term *engineering* in the broadest sense to include both traditional practice in design and manufacturing as well as *computing and information technologies*, and emerging opportunities for those with engineering preparation for practice in medicine, finance, law, business, etc. Thus, their idea could be useful to us in the software engineering or broadly IT profession.

5.1 The I-U-G Roundtable Concept - Original Premises

The formation of the Roundtable was based on two basic ideas [12]:

1. **The need for Actions** rather than further recommendations. The body was primarily designed to establish a viable ***process*** to deal with identified and additional issues, concerns and changes in priority that would inevitably arise as progress was made and the future unfolded;
2. **The prominence of the Industry**:

   Though operating in a true working together spirit among representatives of industry, academia,

government, etc., the Roundtable was **industry-led**. The reasons given include [12]:

- A substantial majority of engineering graduates are employed in professional practice. Industry, as the prime employer, should know better than anyone else what it needs and how technology will be applied
- Industry has the responsibility to take a leadership role in defining and acquiring what it needs to serve its stakeholders and remain viable in the global market.
- The engineering education system benefits from a knowledgeable industry perspective that constructively contributes to reform initiatives by other stakeholders in academe and government.

Since its inception in 1995, the Roundtable has organized several meetings hosted by different companies. Memorandum of Understanding that amounted to a written handshake agreement had been written. Action teams have been formed and activities have included preparation of several technical papers. Significant among these writings were efforts to describe in some specificity what industry needs in terms of university preparation of undergraduate engineering students. An IUGREEE vision of engineering practice as foreseen at an arbitrarily selected time horizon of the year 2010 is currently being validated on an industry-wide basis.

5.2 IUGREEE Objectives

The specific objectives of the IUGREEE are to [12]:

- Articulate and draw attention to critical issues that will affect engineering education as perceived from an evolving industrial perspective.
- Develop action agendas to accomplish needed reform in engineering education
- Facilitate implementation of these agendas through existing organizations and mechanisms i.e. use existing resources to the maximum degree possible.

5.3 IUGREEE Organization and Operation

The IUGREEE is made up of three components, *Action Teams*, a *Policy and Steering Team* and *a Council*, which interact with industry, universities and other organizations (government agencies, professional societies, etc.) as required. The IUGREEE remains a voluntary affiliation of individuals representing their various organizations and institutions operating under a simple agreement to support the aims of the organization. Membership is free. The IUGREEE membership meets twice each year at various sites selected by a hosting organization (company, university, etc.) to report on the status of current activities and to coordinate planned future efforts.

We believe that the formation of this type of Roundtable will go a long way to assist in achieving the goal of encouraging and establishing better collaboration between the academia and industry that will help to combat with some of the main problems confronting our nation - including poor funding, lack of facilities, slow pace of industry growth due to shortage of innovative ideas, etc.

6.0 Recommendations for Stakeholders

- The current Student Industrial Work Experience Scheme for tertiary institutions can be made attractive to Industry partners by introducing some tax relief for every student that an organization employs during the programme.
- The establishment of a body that will be responsible for collaborations between academia and industry. For example the Manufacturing Association of Nigeria (MAN) can facilitate a forum for academia/industry.
- Departments at every tertiary institution could create a database of Alumni.
- Extending the obligations of CPN/NCS/ITAN in enhancing collaboration between academia and industry.
- The setting up of dedicated committees or groups who would liaise with Research and Development departments of Organizations.
- Joint Training/Conference of academia and industry.
- Need to form working groups that will be responsible for collaboration issues

7.0 Conclusion

There are significant benefits derived from the collaboration of universities and industry aimed at meeting the professional development education and training requirements of software engineers. Many universities and industries in the developed world have

benefited immensely from their established collaborations. University lecturers are given much-needed exposure to practical applications and industry trends. They, in turn, enhanced their development, equipment procurement, and other resources for research and consulting. The industry also gained a lot from such relationship such as technology transfer, employment of skilled labour (graduates) etc. We therefore hope that a number of our recommendations will be observed. Just like the IUGREE stated, we need immediate ACTION rather than unimplemented recommendations due to biases or other reasons.

References

Beckman K. (1997). **Directory of Industry and University Collaborations with a Focus on Software Engineering Education and Training, Version 6**. SPECIAL REPORT, CMU/SEI-97-SR-018 Computer Data Systems, Inc. November.

Davies R.M. (1996). **Industry-University Collaborations: A Necessity for the Future.** Journal of Dentistry. Vol. 24, Issues 1-2, Jan-Mar. pp 3-5.

Doutriaux J, Padmore T., Schuetze H.G. (2003). **Conflicts of Interest Arising from Industry-University Interactions.** Faculty of Education, University of British Columbia, Vancouver. October.

Engell S. **Research Collaboration Between Academia and Industry - A report from Germany**. Process Control Laboratory, Department of Chemical Engineering, Universität Dortmund, Germany.

Gerritsen, F.A (1999). **Collaboration between Industry, Academia and End-Users.** Fourth Mayneord Phillips Summer School on Medical Image Processing, Visualisation and Analysis, Oxford. July12-16. pp1-12

Heidi J.C. Ellis, Nancy R. Mead, Ana M. Moreno, Stephen B. Seidman**. Industry/University Software Engineering Collaborations for the Successful Reeducation of Non-Software Professionals**.

Heidi J.C. Ellis, Nancy R. Mead, Ana M. Moreno, Stephen B. Seidman**. Reeducation to Expand the Software Engineering Workforce: Successful Industry/University Collaborations.** SPECIAL REPORT, CMU/SEI-2002-SR-001. July 2002

Industrial Research Institute (1997), Government-University-Industry Research Roundtable, Council on Competitiveness. *Industry-University Research Collaborations: Report of a Workshop (Free Executive Summary)*. National Academies Press. <http://www.nap.edu/catalog/5579.html>

Jones J. (2005). **Creative Research & Development Collaborations.** Sixth Australasian User Interface Conference (AUIC2005), Newcastle, Australia, Conferences in Research and Practice in Information Technology, Vol. 40. *Mark Billinghurst and Andy Cockburn*, (Ed.). Australian Computer Society, Inc.

Kim S (1997). **The Impact of University-Industry Collaborations on Academic Values**. MPA thesis, Faculty of Graduate Studies, Iowa State University Ames, Iowa.

Kornecki A.N., Khajenoori S., Gluh D. and Kameli D. (2003). **On a Partnership between Software Industry and Academia.** Proceedings of the 16th Conference on Software Engineering Education and Training. IEEE.

McMasters J.H., White B.J. and Williams K.A. (1999) **Industry-University-Government Roundtable for Enhancing Engineering Education (IUGREEE)**. March 9.

Mead N., Unpingco P, Beckman K, Walker H., Parish C.L, and George O'Mary**. Industry/University Collaborations: Different Perspectives Heighten Mutual Opportunities**. Journal of Systems and Software. Engineering Institute (SEI) Working Group on Software Engineering Education and Training.

Rao V. and Anderson T. (1999). *Report of the ICEE-1999 International Workshop, Prague, Czech Republic* and *Promoting U.S. Participation In The ICEE-1999, Ostrava, Czech Republic, August 9-14,* - **Final Report** submitted to the National Science Foundation

White C.L., Predebon W.W, Wathne E. and Larsen P.K. (2001). **An International Industry/University Collaboration: Norsk Hydro/Michigan Tech/NTNU**. International Conference on Engineering Education August 6 - 10, Oslo, Norway

Are Public-Private Partnerships the Future for IT Higher Education in Nigeria?

Godwin Ariguzo, Timothy Shea, Rupert Ward

ABSTRACT

This paper reviews Nigeria's current status in the world and within the Sub-Saharan region, examining the Higher Education market, private computer industries within Nigeria, and NIDTA, a national initiative designed to dramatically improve Nigeria's competitiveness in computer industries. The results of a preliminary research study into the current effectiveness of Public-Private Partnerships within Nigerian Higher Education are then discussed. The research study addresses the perceptions of computer industry and Higher Education representatives towards greater competitiveness in Nigeria's IT industry. The survey is presented along with preliminary results. The authors then provide a discussion of further steps which will be undertaken in terms of both continued research, and recommendations for how Nigerian Higher Education Institutions and computer-related business can become a more integrated community in order to make Nigeria more competitive. It is suggested that a Community of Practice would provide an effective method of further facilitating Public-Private Partnerships within Nigerian Higher Education.

Introduction

Since gaining its independence from the United Kingdom about 45 years ago, Nigeria has been controlled for most of those forty-five years by a series of military administrators who have for all practical purposes presided over the precipitous decline of the Nigerian Higher Education system. The woes of the Nigerian educational system from the Primary to Tertiary levels have been well documented (Saint et al, 2004; Dike 2004). In the primary sector, schemes such as the much-lauded Universal Primary Education (UPE) launched in 1976, and the Universal Basic Education plan established by President Obasanjo, have failed to deliver universal primary education to all (Dike, 2004). Between 1991 and 1998 primary school completion rates were between 61 and 77 percent—a "C" performance at best (see table 1). The transition rate from primary 6 to junior secondary was even worse (see table 2). Clearly the goals outlined in the UPE were not being realized.

Table1: Primary School Completion Rates 1991 to 1998, by Gender (percent)

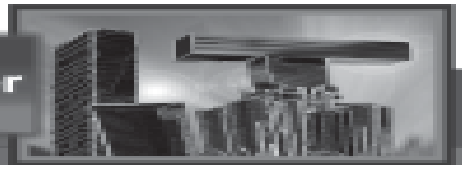
	1991	1992	1993	1994	1995	1996	1997	1998
All	60	70	73	75	69	64	67	65
Male	61	71	72	77	69	63	68	68
Female	59	70	74	72	71	65	64	61

Source: Derived from The World Bank, 2003.

Table 2: Transition Rate from Primary 6 to Junior Secondary School 1 1991 to 1998, by Gender (percent)

	1991	1992	1993	1994	1995	1996	1997	1998
All	48	52	50	48	42	40	40	38
Male	47	55	55	53	46	38	38	37
Female	48	53	53	50	44	42	41	39

Source: Derived from The World Bank, 2003.



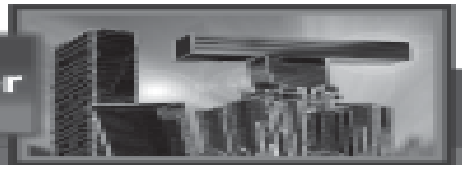
Even the Higher Education system has not faired much better than its primary and secondary education counterparts. Once considered by many experts at home to some of the world's leading universities such as the University of Ibadan (research in tropical health) and Ahmadu Bello University, Zaria (research in agriculture) the Nigerian Higher Education system is now a shadow of its former self. These two were among the six leading universities in Nigeria prior to the oil boom of the 1970's. The pressure to expand university enrollment during the oil boom led to the establishment of many more universities and the subsequent explosion in enrollment. Enrollment grew at an annual rate of 12 to 15 percent in the 1980's and 1990's, from 55,000 in 1980 to more than 400,000 students in 2002 (Dike, 2004). The first generation universities (Ahmadu Bello University, Zaria; University of Ibadan; University of Nigeria, Nsukka; University of Lagos; and Obafemi Awolowo University, Ile-Ife) were all established between 1948 and 1965. Today, 40 public universities and 8 private universities enroll over 400,000 students. With these numbers, Nigeria possesses the largest university system in Sub-Saharan Africa (Saint et al, 2004). In spite of these data, Nigeria with only 4 percent enrollment of university age cohort in Higher Education is comparatively low compared to South Africa at 17 percent, India at 7 percent and Brazil at 12 percent (Saint et al, 2004).

During the various military administrations, many Nigerian scientists and intellectuals left the country to North America and Europe depleting the universities of its research and knowledge creation class. With 20 percent of Africa's population, Nigeria can only boast of 15 scientists and engineers engaged in research and development per million people compared to 168 in Brazil, 158 in India, and 4,103 in the United States (World Bank 2002a).

Gradually, in the past few years, since President Olusegun Obasanjo's election in May of 1999 and re-election in May 2003, it appears that the draconian Higher Education policies of the former military administrators are now being steered in the right direction. Since taking office, the President and his leadership team have embarked on a mission of restoring some of the funds cut from the educational system, albeit at a modest rate. The Obasanjo administration has identified Information Technology as one route the country should take to diversify its economy and become a global player into the next century. It did this by creating the Nigerian National IT Policy and the implementation board The National Information Technology Development Agency (NITDA) in 2001.

NITDA, under the leadership of the late Professor G. O. Ajayi of the Federal Ministry of Science and Technology, was charged with the central mission of making "Nigeria an IT capable country in Africa and a key player in the Information Society by 2005" (NNPIT, 2001). One of the key missions for NITDA was to encourage IT education through partnerships between the private and public sectors. This is a good model that was employed in the United States to bridge the so-called "digital divide" that existed at the beginning of the Internet revolution in that country. One such partnership was discussed in the Report of the Web-Based Education Commission to the President and Congress of the United States (The Web-Based Education Commission, 2000). The report reviews how The Sloan Foundation provided a grant in 1998 to the non-profit Center for Adult and Experiential Learning (CAEL), which in turn worked with the leading telecommunications companies to develop curriculum for New York's Pace University. About this time one of the authors of this paper was awarded $250,000 by Microsoft to help develop a networking certificate program for low-income students and help them obtain personal computers for home use. He was also an advisor to the SBC Corporation and the American Association of Community Colleges on another Public-Private Partnership, developing and delivering high-level computer education programs to community colleges in the United States. We know from experience that these forms of Public-Private Partnerships work.

We believe that that these types of partnerships were what President Obasanjo articulated in his charge. The central questions one must ask, four and half years after NITDA's formation, is what forms of PPPs have been created in Nigeria and did they help achieve the mission outlined in the NITDA mission? Nigeria's future is now, in part, linked with the country's ability to become an effective player in Information Technology related markets such as software development. We believe that for Nigeria to realize this fundamental objective, effective computer education is crucial. However, computer education cannot be conducted in "thin air" and the country's is not yet ready to enter the information age (Pini 1997). Many students lack access to appropriate computer hardware and software in the classroom. It is not uncommon to see IT college graduates equipped with only a theoretical understanding of how computers work and limited hands-on experience. In a study by Mursu et. al, in 2003, "lack of required skills and knowledge" and "lack of skilled personnel" were among the top four risk factors associated with software projects in Nigeria. The general understanding among analysts is that while the financial and oil sectors are the most computerized in the country, educational institutions are in desperate need of attention. To achieve the advances in the



educational system and in the computer industry envisaged in the National Information Technology Policy, strong Public-Private Partnerships (PPPs) are needed.

This paper begins by reviewing Nigeria's current status in the world and within the Sub-Saharan region. Next, the Higher Education market and private computer industries in Nigeria will be examined, along with a more detailed look at NIDTA. A research study is then described to capture the perceptions of computer industry representatives as well as Higher Education representatives in terms of Nigeria's progress towards greater competitiveness. The survey instrument is presented along with preliminary results. Finally, next steps are described that will result in continued research and recommendations for how Nigerian Higher Education Institutions and computer-related business can become a more integrated community in order to make Nigeria more competitive.

Nigeria Today: Information and Communication Technologies in a Global Context and within Sub-Saharan Africa

In order to get some perspective on the current state of Sub-Saharan Africa, there are some useful indices that have been developed and used for a number of years. The Growth Competitiveness Index (GCI), developed for the World Economic Forum, is a macro index that "aims specifically to gauge the ability of the world's economies to attain sustained economic growth over the medium to long term" (Blanke and Lopez-Claros, 2004, p.3). The index has three major components – the environment, public index, and technology index. For 2004-2005, Finland, the US and Sweden are ranked the highest. Of the 103 countries ranked, the top three ranked Sub-Saharan countries are South Africa (41st), Botswana (45th), and Namibia (52nd), while Ethiopia (100th), Angola (102nd), and Chad (103rd) have the lowest ranks. Nigeria ranked 93rd. Table 3 shows the 15 countries with the highest ranking and the rankings of all the Sub-Saharan countries examined.

The Business Competitive Index (BCI), also developed for the World Economic Forum, is a more micro oriented index and thus a complementary index to the GCI for comparing the competitiveness of countries. The BCI focuses not on country factors such as fiscal policies and social conditions but rather on the "sophistication of the operating practices and strategies of companies as well as the quality of the microeconomic business environment in which a nation's companies compete" (Porter, 2004, p. 19). The premise for the index is that the broader, nation-wide infrastructure issues are just the beginning. Individual companies must still act prudently in order to take advantage of the foundation a nation provides.

For 2004-2005 the top three ranked countries are the US, Finland, and Germany. Of the 93 countries ranked, the top three ranked Sub-Saharan countries are South Africa (25th), Namibia (49th), and Botswana (60th) while Tanzania (83rd), Mozambique (87th), and Ethiopia (91st) have the lowest ranks. Nigeria was not ranked. Table 3 shows the 15 countries with the highest ranking and the rankings of all the Sub-Saharan countries that were ranked.

The Networked Readiness Index (NRI) is a model that describes "the degree of preparation of a nation or community to participate in and benefit from ICT developments" (Dutta and Jain, 2004). As shown in Table 3, the US, Singapore, and Finland are the top three. Of the 102 countries ranked, the Sub-Saharan countries that ranked best were South Africa (37th), Botswana (55th), and Namibia (59th), while Angola (99th), Haiti (100th), Ethiopia (101st) and Chad (102nd) have the lowest ranks. Nigeria ranked 79th. Table 3 shows the 15 countries with the highest rank and the rankings of all the Sub-Saharan countries that were ranked.

Nigerian Computer Industries

NITDA

The National Information Technology Development Agency (NITDA) was established in April 2001 to help implement the Nigerian National Information Technology Policy created in March 2001 as an act of the Federal Executive Council. NITDA's vision statement was "to make Nigeria an IT capable country in Africa and a key player in the Information Society by the year 2005, using IT as the engine for sustainable development and global competitiveness" (NNPIT, 2001). The general objectives that relate directly to education include:

- To empower Nigerians to participate in software and IT development
- To empower youth with IT skills and prepare them for global competitiveness
- To integrate IT into the mainstream of education and training

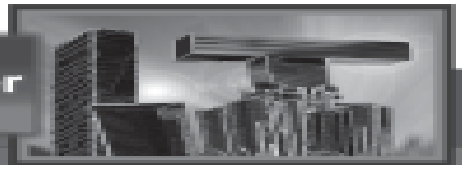
Several general objectives related to the role of the private sector, include:

- To create an enabling government and facilitate private sector (national and multinational) investment in the IT sector

Table 3: Country Indices

Growth Competitiveness Index (GCI) Rankings (macro, country analysis) 2004 – out of 103 (a)		**Business Competitive Index Rankings (micro, business level analysis) -- out of 93 (b)**		**Network Readiness Index Rankings – out of 102 (c)**	
Top 15		**Top 15**		**Top 15**	
1	Finland	1	US	1	US
2	US	2	Finland	2	Singapore
3	Sweden	3	Germany	3	Finland
4	Taiwan	4	Sweden	4	Sweden
5	Denmark	5	Switzerland	5	Denmark
6	Norway	6	UK	6	Canada
7	Singapore	7	Denmark	7	Switzerland
8	Switzerland	8	Japan	8	Norway
9	Japan	9	Netherlands	9	Australia
10	Iceland	10	Singapore	10	Iceland
11	UK	11	Hong Kong	11	Germany
12	Netherlands	12	France	12	Japan
13	Germany	13	Australia	13	Netherlands
14	Australia	14	Belgium	14	Luxembourg
15	Canada	15	Canada	15	United Kingdom
Sub-Saharan Countries Ranked					
41	South Africa	25	South Africa	37	South Africa
45	Botswana	49	Namibia	55	Botswana
52	Namibia	60	Botswana	59	Namibia
68	Ghana	61	Kenya	71	Tanzania
75	Gambia	62	Ghana	74	Ghana
78	Kenya	75	Zimbabwe	**79**	**Nigeria**
79	Uganda	77	Malawi	80	Uganda
82	Tanzania	80	Madagascar	81	Senegal
83	Zambia	82	Algeria	82	Gambia
87	Malawi	83	Tanzania	83	Cameroon
88	Mali	87	Mozambique	84	Kenya
92	Mozambique	91	Ethiopia	85	Zambia
93	**Nigeria**			88	Malawi
96	Madagascar			92	Madagascar
99	Zimbabwe			95	Zimbabwe
100	Ethiopia			96	Mali
102	Angola			97	Mozambique
103	Chad			99	Angola
				101	Ethiopia
			Nigeria (not ranked)	102	Chad

(a) Blanke and Lopez-Claros, 2004 (b) Porter et al., 2004 (c) Dutta and Jain, 2004



- To stimulate the private sector to become the driving force for IT creativity and enhanced productivity and competitiveness

For (NITDA), 2005 is the year to assess how far the Information Technology industry has come in Nigeria. The next sections describe the results of the preliminary data collection and a research plan for further investigation.

Research

Research Context

Studies addressing the level of Public-Private Partnerships involving IT in the mainstream of education and training are either dated or limited. This paper, based on a review of current literature and a preliminary questionnaire-based survey of prominent software companies and program leaders in Higher Education Institutions in Nigeria, outlines the findings to date as well as provides a discussion of how this research study might be extended and suggestions for future development in the IT sector.

Data Collection

In addition to a literature search, two questionnaires were developed – one for prominent software companies in Nigeria and one for program leaders in Higher Education Institutions in Nigeria. The questions targeted three areas:

- the amount and nature of private/public partnerships currently undertaken in Nigeria;
- overall perceptions of the current strengths and needs of Nigeria's Higher Education institutions related to Information and Communication Technologies (ICT);
- overall perceptions of Nigeria's progress over the past four years, as a nation, towards the NITDA goals.

A condensed version of the questionnaire is provided in Appendix A. Four Nigerian computer companies, one Higher Education Institution in Nigeria, and a representative from the National University Commission completed the questionnaire during April 2005. Three of the four companies had annual sales of greater than 50,000,000 Naira, ranging in number of employees from 6 to 100 with about two-thirds of their employees full time. The companies had a range of zero to 37 recent hires from at least two-year degree institutions. The Higher Education institution was large, with over 10,000 students.

Preliminary Findings

At the moment, there a currently some ICT partnership programs between industry and Higher Education Institutions in Nigeria. Three of the four companies were active and generally satisfied with their efforts. Initiatives included using student fees to support an ASP (Application Service Provider) model for students' technology infrastructure or for providing hardware and software solutions. Three of the four companies planned to increase their ICT partnerships with Higher Education over the next two to three years. The Higher Education Institution also planned to increase their ICT partnerships with industry.

The greatest strengths in terms of ICT for Nigeria's Higher Education Institutions and academic programs that were identified included the quality of the people in both school and industry and proven partnering strategies. Current needs included greater participation by industry giants such as Oracle and Microsoft, training for faculty as well as affordable and reliable high speed Internet. Current ICT skills most needed by Nigerian companies today were identified as software development – especially Internet and e-business development – and networking.

Progress over the past four years towards the goals established by NITDA was perceived as ranging from modest to moderate, by the computer industry representatives, the Higher Education Institution, and the National University Commission (using a 10 point scale where 1=poor and 10=outstanding). Table 4 shows how the respondents rated the progress made on the five key goals of NITDA.

The next section describes, based on the information collected to date, a more comprehensive research plan

Table 4: Ratings of the Progress Made on the NITDA Goals

(=N=6)	2001	2005
° To empower Nigerians to participate in software and IT development	1.2	3.1
° To empower youth with IT skills and prepare them for global competitiveness	1.2	3.0
° To integrate IT into the mainstream of education and training	1.2	2.8
° To create an enabling government and facilitate private sector multinational) investments in the IT sector	1.2	3.2
° To stimulate the private sector to become the driving force for IT creativity and enhanced productivity and competitiveness	1.2	3.3

to better understand the current situation as well as make useful recommendations for the future.

Conclusions and Next Steps – Community Development (CoP)

The next steps in the research plan include: (1) using the existing questionnaires to collect more comprehensive data from the Higher Education Institutions and computer industry companies in Nigeria related to Higher Education/industry computer technology partnerships; (2) to conduct a needs analysis in order to create a list of recommendations for improving such partnerships; (3) to explore the possibility of creating a Community of Practice (CoP) specifically targeted to facilitating such partnerships.

1. Collecting More Primary Data

To date, we have collected completed questionnaires from four ICT companies in Nigeria, one Higher Education school in Nigeria, and a representative from Nigeria's National University Commission. In the second round of data collection, all the Higher Education Institutions in Nigeria will be sent a questionnaire through e-mail and surface mail. Currently there are about 40 public universities and 8 private universities, as well a number of other tertiary institutions of Higher Education in Nigeria. Our goal is to encourage as many of as possible to participate in the study. The relevant computer industry companies will be identified by turnover and staffing. Follow-up e-mails and phone calls will be used to attain our target of a fifty percent response rate for each group. Parties attending the Nigeria Computer Society 8th International Conference in June 2005 will be asked to contribute to this data collection where appropriate.

2. Analysis

The questionnaire results will be analyzed to determine recommended steps and priorities in order to facilitate effective Public-Private Partnerships for IT Higher Education in Nigeria. In particular, the results will be reviewed as a primary design input for developing a Community of Practice (CoP) focused on facilitating these Public-Private Partnerships.

3. The Potential of a Community of Practice (CoP)

The solution to facilitating Public-Private Partnerships that will improve IT Higher Education in Nigeria will likely be a multi-faceted one that will benefit from government initiatives, international company initiatives, local company initiatives, school initiatives, individual faculty initiatives, and more. Two common problems for any one of these groups as they try to move forward is in finding the right mix of information and in contacting the right person to address specific research questions. A web portal, focused on supporting the needs of the different people looking to get involved in Public-Private Partnerships, would provide an effective means of facilitating the information content and connectivity needed. Over time, it is expected that both the information and connectivity needs will evolve as the needs of the participants evolve.

In a more formal sense, such a portal is referred to as a Community of Practice (CoP). Allee (1997) described a CoP as individuals who are informally bound to one another through exposure to a similar set of problems and a common pursuit of solutions. Wenger & Snyder (2000) have applied the concept to the Internet age by focusing on the potential of online communities. An online CoP, typically implemented as a web portal, has three main characteristics:

- *The domain:* A CoP is not just a group of friends, involvement requires some knowledge in a specific field, or domain.
- *The community:* Members of the community interact and learn together.
- *The practice:* Members of the community develop over time as they individually and collectively solve problems and communicate with each other. As such, the web portal needs to evolve as well in order to continue to serve the community as the community's needs change.

In this case, the domain would be people who are interested in developing or sustaining Public-Private Partnerships that will improve IT Higher Education in Nigeria. The community would be the people from all the different interest groups who participate. The practice would be the collective evolution of the community as they become more practiced at partnering and as various infrastructure and environmental components evolve.

As an example of what part of such a CoP might look like, consider the needs of a faculty member who wants to update his or her technical skills. As part of the CoP to support Public-Private Partnership for improving IT Higher Education in Nigeria, the CoP might include a section that highlights information and connectivity from a wide array of sources: university and vendor workshops and course announcements, technology company websites and contacts, publishing company

websites and contacts, professional associations including listserves, funding opportunities to pay for continuing education, models for train-the-trainer programs, and more (Sherer, et. al., 2002).

By including the features identified from the data analysis, the CoP concept will be an effective means for facilitating Public-Private Partnerships to help IT Higher Education in Nigeria. We are interested from both a practical and research perspective in bringing the best ideas to bear on developing and sustaining a Public-Private Partnership web portal that supports anyone interested in creating or fostering such partnerships.

References

Allee, V. *The Knowledge Evolution: Expanding Organizational Intelligence.* Boston: Butterworth-Heinemann. 1997.

Blanke, Jennifer and Augusto Lopez-Claros (2004), "The Growth Competitiveness Index: Assessing Countries' Potential for Sustained Economic Growth", *The Global Competitiveness Report*, World Economic Forum, Palgrave MacMillan, New York.

Dike, Victor (2004), "The state of education in Nigeria and the health of the nation," http://www.africaeconomicanalysis.org/articles/gen/education10204234737htm.html, published February 2004, retrieved April 18, 2005.

Dutta, Soumitra, and Amit Jain (2004), "The Networked Readiness Index 2003-2004: Overview and Analysis Framework", *The Global Information Technology Report,* World Economic Forum, Oxford University Press, New York.

Mursu, A., K. Lyytinen, H. A. Soriyan, and M. Korpela. "Identifying software project risks in Nigeria: an international comparative study," *European Journal of Information Systems*, Basingstoke: Sep 2003. Vol 12, Iss. 3; pg.182.

National University Commission (2005), "Universities in Nigeria" http://www.nuc.edu.ng/Universities/Universities.htm, retrieved April 18, 2005.

NNPIT's "Nigerian National Policy for Information Technology IT)," http://www.nitda.org/docs/policy/ngitpolicy.pdf, published March 2001, retrieved January 25, 2005.

Pini, A., "Training on thin air," *African Business*, London: Mar 1997. Iss. 219; pg. 38.

Porter, Michael, Klaus Schwab, Xavier Sala-i-Martin, and Augusto Lopez-Claros (2004), *The Global Competitiveness Report*, World Economic Forum, Palgrave MacMillan, New York.

Saint, William, Teresa A. Hartnett, and Erich Strassner (2004), "Higher Education in Nigeria: A Status Report," World Education News and Reviews, http://www.wes.org/ewenr/PF/04sept/PFFeature.htm, retrieved April 18, 2005.

Sherer, P., T. Shea, & E. Kristensen. "Harnessing the potential of online faculty development:

Challenges and opportunities." In D. Lieberman (Ed.), *To Improve the Academy,* 20, 162-179. Bolton, MA: Anker.

Web-Based Education Commission (2000), "The Power of the Internet for Learning: Moving from Promise to Practice" Report of the Web-Based Education Commission to the President and Congress of the United States, Washington, DC.

Wenger, E., & W. Snyder. "Communities of practice: The organizational frontier.", *Harvard Business Review*, Jan.-Feb., 139-145.

World Bank (2002a), In Saint, William, et al., (2004), "Higher Education in Nigeria: A Status Report," World Education News and Reviews, http://www.wes.org/ewenr/PF/04sept/PFFeature.htm, retrieved April 18, 2005.

World Bank (2003) "School education in Nigeria: Preparing for universal basic education - A Country Status Report." Washington, D.C. http://wwwwds.worldbank.org/servlet/WDS_IBank_Servlet?pcont=details&eid=000012009_20040419105714, retrieved April 18, 2005.

The Next IT Society

Osuagwu, O. E, Ezeji, M.

Abstract

This paper looks at the past, present and future of IT diffusion. We have tried to forcast what shape all spectra of IT will take at the beginning of the 21st century and requests the Nigerian IT Society to be ready to face new challenges if Nigeria must join the emerging global Information Society.

Background of Study

This research is precipice of three things. Firstly, the unparallel speed in which information system has evolved. Secondly, is the objectivism of what Ian Miles called information revolution. Finally, is what we are witnessing as espoused by the great management grandmaster Peter F Drucker in "The Next Workforce". Drucker espoused that the next workforce (which we are experiencing already) will be knowledge-based. Hardware, software, mindset and concepts of information processing[1] are evolving. The pace of change brought about by (new) technologies has had a significant effect on the way people live, work, and play worldwide. New and emerging technologies challenge the traditional process of doing virtually everything including curricula of any field of study. Easy communication provides instant access to a vast array of data, challenging assimilation and assessment skills. Rapid communication, plus increased access to IT[2] in the home, at work, and in educational establishments, could mean that learning becomes a truly lifelong activity—an activity in which the pace of technological change forces constant evaluation of the learning process itself.

Towards late 20th century, changes in taste with subsequent production and use of information become too pronounced. Information, hence information specialist and technology have been there as old as creation. Two things underpin this late 20th C turn. First is the social and organizational change. This calls for extra information processing capacity to enable decisions to be taken. The second is tech change! The ***new*** information technology (IT) is based on ***microelectronics***, together with other innovations such as optical discs and fibre optics. This underpins huge increases in the power, and decreases in the costs, of all sorts of information-processing. The term "information-processing" covers the generation, storage, transmission, manipulation, and display of information, including numerical, textual, audio, and video data. The information-processing aspects of ***all*** work can be reshaped through IT. Computing and telecommunications (and also such areas as broadcasting and publishing) used to be quite distinct industries, involving distinct technologies. Today, they have converged as ICT[3] around certain key activities, such as use of the Internet. Using the same underlying technologies, modern computing and telecommunication devices handle data in digital form. Having established the reason cum background of this study, lets look at journey today, history.

Brief History

The invention of movable type in the mid-15th century and the creation of the portable typewriter at the end of the 19th century are but two landmarks. Each of these inventions led to a profound revolution in the ability to record and disseminates information. The first large-scale mechanized information system was Herman Hollerith's census tabulator. Invented to process the 1890 U.S. census, Hollerith's machine represented a major step in automation, as well as an inspiration to develop computerized information systems. One of the first computers used for such information processing was the UNIVAC I, installed in the U.S. Bureau of the Census in 1951 for administrative use and in General Electric in 1954 for commercial use. Beginning in the 1970s, personal computers brought some of the advantages of information systems to small businesses and to individuals, and the invention of the World Wide Web in the early 1990s accelerated the creation of an open global computer network. This was accompanied by a dramatic growth in digital human communications (e-mail and electronic conferences), delivery of products (software, music, and movies), and

business transactions (buying, selling, and advertising on the Web).

The Now: Impacts of information systems

In Organization

Today, information systems bring new vistas to the way companies interact, the way organizations are structured, and the way workplaces are designed. In general, use of network-based information systems significantly lower the costs of communication among workers and firms and enhance coordination on collaborative projects. This has led many organizations to concentrate on their core competencies and to outsource other parts of their value chain to specialized companies. The capability to communicate information efficiently within a firm has also led to the deployment of flatter (lean) organizational structures with fewer hierarchical layers.

"Virtual" organizations have emerged that do not rely on physical offices and standard organization charts. Two notable forms are a network organization and a cluster organization.

In a network organization, long-term corporate ***partners*** supply goods and services to and through a central firm. Together, a network of small companies can present the appearance of a large corporation. Indeed, at the core of such an organization may be nothing more than a single entrepreneur supported by only a few employees. Thanks to information systems, product specifications in an electronic form can be modified during computerized video conferences between employees throughout an organization—after which supplies can be secured and distribution coordinated, using automatic electronic forms as sales orders are received. Wide area networks and the Internet in particular, help partnering organizations to facilitate the interaction of widely dispersed business units.

In a cluster organization, the principal work units are permanent and temporary teams of individuals with complementary skills. Team members, who are often widely dispersed around the globe, are greatly assisted in their work by the use of intranets and groupware.

Information systems built around portable computers, mobile telecommnications, and groupware have enabled employees to work not just outside the corporate offices but virtually anywhere. "***Work is the thing you do, not the place you go to,***" has become the slogan of the emerging new workplace. Virtual workplaces include home offices, regional work centres, customers' premises, and mobile offices of people such as insurance adjusters. Employees who work in virtual workplaces outside their company's premises are known as telecommuters.

In Organizational Structures

The "hollow firm" is one effort to gain (more) flexibility. The company attempts to dispense with the direct ownership and operation of many facilities that would traditionally have belonged to it, instead outsourcing production, distribution, and other tasks to other firms. Many computer companies, for example, buy in many or most of their components from specialist suppliers, and some firms do little more than design the computer for others to assemble.

A related idea is "de-layering", or "flattening", in which the company tries to do away with the numerous layers of middle management and administration that have traditionally processed information, and communication flows between the senior staff and the shop floor or fieldworkers. New information systems are typically used to allow rapid communication across a reduced number of organizational layers.

In Employment

In the late 1970s and early 1980s, when word processing first began to be taken up on a large scale, massive job losses were witnessed. However, this is no reason to assume that existing structures will endure. Industrial interest in new forms of organization, such as novel management structures, coordination of activities over large distances by means of telecommunications, teleworking, and other forms of distance working, indicates willingness to consider change. Whether a loss of clerical jobs will result remains much debated. Some commentators point to job losses in office-based sectors such as financial services, which use IT intensively, as a harbinger of things to come. While some office jobs may have gone, some other traditional clerical jobs have been upgraded to involve new functions made possible with new IT, such as desktop publishing, database management, and customer services.

In Communication

By the late 1990s, the integration of office IT become apparent: material increasingly exchanged by e-mail (which has finally established itself); many professionals use personal computers directly, often at home and while traveling, as well as in the office; and increasingly, personal computers are networked.

and/or digitized.

In Consumerism

At different rates IT is diffusing into the home. The implications of consumer innovations can be substantial. Widespread use of cars facilitated new ways of life, with a growth of suburban living and out-of-town shopping centres, and a decline of train and bus services. The expansion of consumer IT is associated with changes in ways of working (for example, telework), playing (new home entertainment systems), shopping (teleshopping), and learning (multimedia products of various sorts, such as CD-ROMs, online lectures, e-learning).

In Medicines

IT can be used in monitoring body conditions (digital thermometers, pulse meters, and blood-pressure meters are available), and in providing health and lifestyle monitoring and advice (recommending exercise levels, medical check-ups, or diets). Telephone help lines have long offered advice, counseling, and medical services; these and many other services are beginning, sometimes in rudimentary form, to be provided on the Internet. Technology aided surgery is being given attention.

Tomorrow: The Next IT Society

It is very difficult to say what tomorrow will look like. It is above the rein of moltal man. Whatever we will guess as being the trend of the next IT society is basically as a result of ongoing researches and present breakthroughs. Many professions and ways of doing things will drastically change from the present status as in few of the following ways;

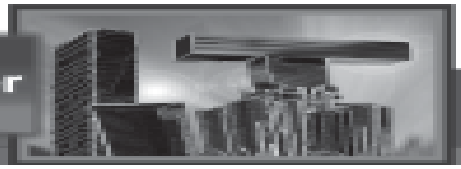
Communications: It is amazing to know the levels of innovations going on in telecomm world. Email address interface with GSM SMS, Email to fax, e-banking giving way to m-banking. We have heard of e-banking. Now is m-banking. If in the next IT society we are going to enjoy popularized strong and reliable m-banking (mobile banking), the next society will redefine tradition modus operandi of work and modus vivendi of telecom equipment. Telephone handset will be used in self service bank activities like checking of account balances, transferring of one account to another, etc. Powerful handset will redefine mobile computing and remove it from the premise of palmtops and notebooks to telephone handsets. The also camera coys are in trouble as they must adapt to infusing camera in handset as the next cameras will incorporate handsets

Journalism: Journalism in Nigeria is practiced as the journalist goes about with walkman to pick interviews and comes to office for transcription. This will become old fashion as newspaper and TV journalism will combine communication and microelectronics (ICT) in future. The computer will scan the newspapers looking for topics of interest and will check the advertisements looking for items that we have indicated to it to print.

E-Books: Online e-books are the fastest selling proprietary rights now. Books that must sell must be e-books (virtual) heavily loaded with hyperlinks. This is even the fastest link to covering heavy volumes. Though prints will be in great demand, e-book will more.

Secretarial Job: Taking dictations with shorthand and transcribing, typing with typewriter and word processor are already old fashioned. Email (electronic mail) and EDI (electronic data interchange), net meetings are in vogue. This is used in business context today, but there is no reason why EDI techniques cannot be used by consumers as well. Typesetting aspect of secretary's job will be every person's work in future. Real secretarial jobs will reside in managing the office and the boss, eyes on the scheduler. Next society will have full voice recognition that will translate voice to typeset using microphone. Now it is not perfect.

Work: The development and subsequent development of Telediagosis and Teleworking technology will eliminate the constraint imposed by distance and time in effective implementation tasks where there is a dearth of professional personnel. It is has being said earlier on that work is what you do not where you are. Telediagnosis uses microprocessor chips and standard communications facilities to allow a device such as an appliance or a computer to automatically place a telephone call to a diagnosis centre. The IBM 3090 computer, for example continually monitors its own operation. If it detects a problem, the system places a telephone call to a diagnostic centre or helpdesk. The engineers can examine the data, request other information, and essentially operate the IBM 3090 in diagnostic mode. Assuming the problem is not disabling the computer, normal data processing can continue while the diagnostic work is underway. In a high percentage of cases, the remote diagnostic centre can determine which part of the percentage of cases is falling and can dispatch a local technician with correct replacement part. This concept could just as well be implemented in home appliances such as televisions, washing machines, or furnaces. Similarly, an automobile might be driven to the local dealer's garage, where it could be connected to a diagnostic computer that



communicate with a central diagnostic computer at the auto manufacturer's headquarter. The techie exists already. Electric computerized car has been produced in Japan. It doesn't need fuel. In future, the economics will drive the general availability of such services. The techie could be adopted for a tele-worker in maintenance engineering coy.

Medicine: The dearth of medical doctors in remote communities has caused the death of many Nigerians. The evolution of medical telediagnosis is going to eliminate this problem in the next society. Here a computer could give doctor recommendations about how to access a particular patient complaint. Using audio and video signals and diagnostic medical equipment with communications capability – such as a communicating x-ray machine – a doctor at one location could converse with a patient in another location, conduct tests, examine the results, diagnose the condition, and prescribe a treatment. The telecommunications connection would allow patients in remote geographic location to consult with specialist in medical centres and provide doctors the capability to consult with one another and share info about particularly unique or difficult medical cases. With the technology one doctor can cover a large geographical area with high degree of efficiency and with less hassle. In the next society, surgery can be performed with a lot of expertise supervision online.

Security [Policing]: Nigeria don't have computerized police. Go to police station 100 times they will give you piece of paper to write your case/statement. Osuagwu (2001) recommended the application of Automobile Navigation by Satellite (ANS) which is a car equipped with a small television-like device with capability to displaying a map of local area. A transmitter in the car woluld sends a signal back to the car. A microprocessor in the car translates the signal from the satellite and display the location of the automobile on the map. Ocean-going ship have been using a similar technique with Inmarsat satellites for years, but until now the equipment has been expensive and satellite capacity far too limited to make the technology available to consumers. With the increasing bandwidth of a satellites and the decreasing cost of the electronics required in automobiles, the application is coming closer to being affordable in the consumer market. This technology could act as excellent teleworking tool for security agencies and the police. The future holds lots of prospects for consumer application of this technology.

Smart Homes: Smart homes are all about controlling home functions such as heating, lighting and security alarms using microprocessors. When coupled with telecommunication, the computer can be instructed to call home just before leaving the office. It could be used to call someone for lunch. Already a robot is being used to provide jokes to one lonely ones and to entertain as in sports arena. In the near next society, the robots shall be more enhanced and commercialized. A smart home might also contain smart appliances that have built in diagnostic capability. E.g. in this case are Xanadu House in Kissimmee, Florida and Ahwatukee House in Arizona which are designed to be energy efficient and computer controlled. The smart home of the future will have a common wiring system.

Conclusion

The outcome of the IT spearheaded revolution depends on social action and choices as well as on technological developments. Just as industrial societies around the world take various forms, and there are very different ways of life within them, so it is likely that there will be a wide range of information societies. However, as new IT permits more global communication, and more firms expand into global markets, there are also strong forces at work to share elements of different cultures around the world on an unprecedented scale. It was Bell (2004) that said that however disturbing this challenge may at first seem, the nature of technology is that it not only poses problems but also offers solutions—constantly creating opportunities and providing new and creative solutions to the process of living and learning. The speed is too fast. The only way to survive it is to have a regular revisable curriculum.

References

Asa Briggs, Technology and the Media, contained in Britannica Encyclopedia (2004)

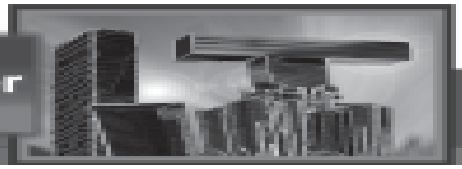
David G. Messerschmitt, Understanding Networked Applications: A First Course (2000),

Ian Miles, Information Revolution, contained in Microsoft ® Encarta ® Premium Suite (2004)

Manuel Castells, The Information Age: Economy, Society and Culture, 3 vol. (1996–98), is a broad analysis of the international information society.

Nicholas Negroponte, Being Digital (1995); and Michael L. Dertouzos, What Will Be: How the New World of Information Will Change Our Lives (1997).

Osuagwu, O. E., Information Technology Diffusion and Employment Opportunities in the new Millienmium, contained in Journalism of Professional Administration, January – March 2001



Rob Kling (ed.), Computerization and Controversy: Value Conflicts and Social Choices, 2nd ed. (1996)

Vladimir Zwass, Foundations of Information Systems (1998).

Zachary Cohn and Simone Lafolii, Mini Illustrated Dictionary of Information Technology, 1995.

Human-Computer Interaction factors: Usability and User experience in Software Designs

Olubukola D. Adekola, Joshua O. A. Ayeni

ABSTRACT

The subject of human-computer interaction factors, usability and user experience is meant to bring about a shift in the way software systems are designed and delivered to users. For every software, be it system or application software, there exists a user. How this user is thought of should be evidenced in the design. At this juncture, it should be noted that thinking about the user connotes a design for the user. A software that passes through this phase would not leave the user hanging or wandering as to what to do next at any point, i.e. it will always provide a route, a path or a way out. The focus here is not just a software that would function, of course one should expect a software to function or serve its purpose. The question is: how is it serving that function? This paper stresses and justifies the need to bring usability and user experience factors into today's design. Interaction design represents a system that makes use of most of human skill, judgement and support rather than a system that constrains the user as in software engineering where focus is mainly on the production of software solution for the given application.

Introduction

For the past twenty years, technological developments have enabled computers to pervade most aspects of human life to the extent that almost everyone comes in contact with computers in one way or the other. Today, the range of knowledge and experience of different users are very broad. Therefore, it is important that the way in which people interact with computers is intuitive and clear. So, bringing interaction design into our design processes and principles is crucial and relevant. This has profound influence on both individuals and organisations and lack of attention to human-compure-interaction (HCI) can endanger life in safety-critical situations. Concisely, the goals of HCI are to produce usability, user experience, safety, as well as, functionality in systems.

Usability, a key concept in HCI, is concerned with making systems easy to learn and easy to use. Poorly designed systems (particularly the interface) can be extremely annoying and frustrating to users. Observation reveals that many users bash their keyboard indiscriminately, even strike their terminal to damage, all a consequence of systems with poor HCI concepts. This subject is not just Software Engineering alone. In order to produce systems with the afore-mentioned qualities, interaction designers, HCI specialists strive to understand factors (such as psychological, ergonomic, social factors, etc.) that determine how people operate and make use of computer technology effectively, and to translate that understanding into the development of tools and techniques to help designers ensure that systems are built to serve people.

The underlying belief in HCI design is people, using the computer systems should come first. People do not have to conform to "fitting in with the system or bending for it" but systems must be built to match their requirements- serve and support users. This work is to present the relevance of the look and feel of software. That is, what system really meet people's need, what system *put understandable face on technology*, and what system would carry functionality not at the expense of enjoyability and ease of use. And what systems are truly user friendly as some user-hostile still carry user friendly logo. To justify this, a review of what has been published was referenced and evaluation of some software and applications was carried out.

Problem Statement

The truth is that the importance of designing for the user cannot be over-emphasised. To do that is to design to touch usability and user experience goals. Unfortunately, a good number of software designers design solely for functionality .i.e. they are only concerned with "designing to make it work". The argument is that even if software system should work,

it must be a delight to the user. Considering Human-Computer factors which evolved from the field of Human-Computer Interaction and Interaction Design are the subject matter of this presentation. What must a developer consider to achieve usability in a software? What should the interface be to achieve a system for the people? What is a simple and friendly interface that does not carry bizarre metaphors as well as icons which confuse and frustrate rather than support? What design would respect Human-Computer Factors and as such respect, accord, honour and serve the users. Since the prototype we used to defend this cause is a web application then the question should be what design would offer usability, user experience and user-friendliness for real?

When do we say a system is successful?

To say many Information Systems fail does not mean that they do not work (function); rather that they fail to meet initial aspirations. Many factors can hinder an Information System from achieving its objectives but the degree of acceptance or rejection by individuals or organizational users is often important.

Gibbs (1994), Scientific American, reported that three-quarters of all large systems are 'operating failures' that either do not function as intended or are not used at all

Why do systems fail?

There is need to understand what constitute success for information systems.

Almost all information systems have a human (user) element and it is often the way in which this user element is handled that can determine success or failure.

Categorically, failure might occur in three ways:

1. Technical failure: This is the level of either a hardware, software or communication fault
2. Utility failure: The IBM Dictionary of Computing (1993) has this to say: utility is the capability of a system, program or device to perform the functions for which it was devised. Failure of this kind might be a consequence of wrong requirement capture.
3. Usability failure: The IBM Dictionary of Computing (1993) defines usability as: the quality of a system, program or device that enables it to be easily understood and conveniently applied by the user. Usability failure might be, most times, due to a poorly designed software interface.

This paper addresses aspects of the third category. The methods in which users are involved might be loosely called human-computer factors. The subject of concern is the need to bear in mind factors as human, environment etc. at every phase of software development.

Interaction Design

Interaction Design is a new and fast growing field that travels beyond traditional industrial design / Software Engineering discipline. Interaction Design comes from an interdisciplinary field of Human-Computer Interaction (HCI) and is concerned with the design, implementation and evaluation of interactive computing systems for human use and with the study of major phenomena surrounding them.

Other fields that are contributory to designing systems to match users' goals are human factors, cognitive ergonomics, psychology, and cognitive engineering. Systems exhibiting good HCI comes from cognitive theory and user research.

Software Engineering versus Interaction Design

Designing products that really meet users' needs and are, indeed, easy to use and enjoy goes beyond employing Software Engineering principles. Again, it is not just thinking what capabilities the system should have. A designer needs to consider the interaction that happens between a user and a computing or communication tool/system. Thinking about software products that people actually enjoy using has an immediate competitive advantage in the market place.

Up to date, developers are not critically addressing issues touching human factors, usability, user experience. Users still have to cope with 'user-hostile' interfaces in the guise of 'user friendly'. While traditional design focuses primarily on a working system (production of software solution for given applications), interaction design invests in understanding people's psychological processes when interacting with tools like computers. Pioneers of HCI in the United State Of America (USA) were primarily concerned with how computers could enrich our lives and make it easier.

Good Interaction Design programs create usability and user experiences that enhance the way people work, communicate and interact.

Design Principles in Interaction Design

These are things designers must consider while conceptualising usability. Design principles are derived from a mix of theory-based knowledge, experience and common sense. They suggest to the designer what to provide and what to avoid at the interface and other parts of the system-the do's and don'ts of Interaction Design (Thimbleby (1990)). They do not specify how to design the actual interface but tell them things they must provide.

Common design principles include:

- Visibility

  Afunction made visible suggests that the user could have likelihood of knowing what to do next. In contrast, when functions are "out of sight", it makes them more difficult to find and know how to use. Norman (1988) describes the controls of a car to emphasize this claim. The controls for different operations are clearly visible e.g. headlights, horn, indicating what can be done.

- Feedback

  Feedback is about sending back information about what action has been done and what has been accomplished, allowing the person to continue with the activity.

- Constraints

  This is making ways of restricting the kind of user interaction that can take place at a given moment. For instance, activity diagram shows which objects are related to which, thereby constrains the way information can be perceived. Norman (1999) classifies constraints into:

  1. *Physical constraints:* This has to do with the way physical objects restrict the movement of things. For instance, the way an external disk can be placed into a disk drive is physically constrained byits shape an size so that it can be inserted in only one way

  2. *Logical constraints:* They rely on people's common-sense reasoning about actions and their consequences. Making actions and their effects obvious enables people to logically deduce what further actions are required. Disabling menu options when not appropriate for the task in hand provides logical constrairt.

  3. *Cultural constraints:* Rely on learned conventions, like the use of red for warning, the use of smiley face to represent happy emotions etc.

- Mapping

  This describes the relationship between controls and their effects in the world. A good example of a mapping between control and effect is the up and down arrows used to represent the up and down movement of the cursor, respectively, on a keyboard.

- Consistency

  Consistency means designing interfaces to have similar operations and using similar elements to achieve similar tasks. A consistent interface is one that follows rules, such as using the same operations to select all objects or doing selection operation with the same procedure. A consistent interface enhances learning.

- Affordance

  This is used to refer to an attribute of an object that suggests how to use object to people. For example, a mouse button invites pushing. Simply, to afford means to " give a clue" (Norman 1988).

Norman clarified that there are two kinds of affordance:

1. *Real Affordance*- belongs to the physical objects e.g. grasping. This is obvious and needs no learning.

2. *Perceived Affordance*- screen-based user interfaces are virtual and one need not try design for real affordances. These are better conceptualized as perceived affordances. These are essentially learned conventions.

Generally, employing HCI concept has brought our attention to building designs that would be usable, easy and pleasurable.

A Review of Related Work

- A Sample Rude Software

  Most digital products are built from an engineering point of view. Consider an example of a rude software which blames the user for making mistakes that are not the user's fault, or should not be. Error message boxes like the one in Figure 1 pop up like weeds announcing that the user has failed yet again. These messages also demand that user acknowledge his failure by clicking 'OK' (www.media.wiley.com).

- A design on "comfort and efficiency" of the user

  Bruce Tognazzini in his 2003 (www.askTog.com) paper has this to say: Simply put, in interaction

Fig. 1: Whose duty is it to notify the library? The program or the user in this context? What is it to even notify? Why tell us this?

design, the user is the prime target. The user cost more than the machine. The concept goes as far as what happens on the user's end even if it will be at the expense of the machine. That is, cheat the machine if that is what it will take to promote user's efficiency. This was illustrated by the instance of conditioning a microwave to heating water for one minute eleven seconds rather than the sufficient one minute ten seconds. Here, the user is favoured in that, if it were one minute ten seconds, the user must press '1' key twice and then visually locate the '0' key and as such the speed of acquiring the last key is dependent on its location which must surely take more than one second to achieve (Fitt's Law -Bruce (2003)). But for the other, the user only need to press same '1' key trice. So, in this context, the longer it is heated the faster. In addition, seeking out a different key not only takes time, it requires a fairly high level of cognitive processing.

Evaluation of some popular software.

- Some metaphor and composite metaphor do not pass the message

  By definition, a mental model is what the mind generates when a person sees or perceives something while interface metaphors are based on conceptual models that combine familiar knowledge with new concepts. This provides users with a familiar orienting device and helps them to understand and learn how to use a system Bizarre use of metaphor builds in user incorrect mental model. The interface tends to fail to give what a user expects after a click on a button. If a metaphor is to be made short, it should not be traded off for comprehensiveness. A balance must be registered at this point. Failure to build correct user's model that can provide help in terms of guiding users behaviour and of course this is the root cause of frustration- resulting in stereotypical "venting" behaviour. The argument is that a user should be able to trace functionality as much as possible from an icon or a metaphor. Check out the flaw in the interface of the *POPULAR* yahoo site (www.yahoo.com) visited by millions of people daily. New user attempting to write and send mails might not know that *COMPOSE* (see fig.2) metaphor is the right answer. Arguably, *COMPOSE* could mean composing a song, a line, a message etc. Instead, *WRITE MAIL* could pose a better composite metaphor just as *CHECKMAIL* is more efficient for what it stands for i.e. you already know where you are heading for.

  Also, this illustration could be likened to a Compact Disc Player which having inserted a disc displays *"READING"* as a form of getting back to the user or feeding him back so as not to lock him in worry. We stand to criticize this by asking the designer and not the system itself that what is the player reading? Could it be the user's mind, the environment, some things about the inserted disc, or the disc itself? This, as many designers, so care-free could truly raise many questions in the minds of the users. And designers would remain ever guilty if they claim to build for the users and yet they still do not take not of though seemingly little but vital things. We need not border the users with *technical terms* or details if truly we claim to build for them. Often times, it confuses rather than informs them. A verbal metaphor like "WAIT" is a friend to the user, expert or novice.

- A typical visual user interface that leaves the user wandering

  The information box below pops up while the user is trying to upload a file on our favourite hotmail message site (www.hotmail.com). Funny enough the software does well to beg the user (see fig.3) by displaying the message " *Please wait while we upload your file*" but has completely lost sight of informing us of the extent of task either with a gauge bar or percentage or any thing at all Interaction design places values on informing the users and giving appropriate feedbacks. Regrettably, hundreds of our today's software are rather user-hostile than user friendly.

An Example of a Ticketing System

This design is a prototype that addresses some of the user and usability issues. It is to create a correct model in the mind of users. The key subject of this prototype is the design and implementation of a web-based ticket reservation system that would emphasis the relevance of interaction design across software development both web and non-web applications. It presents basically

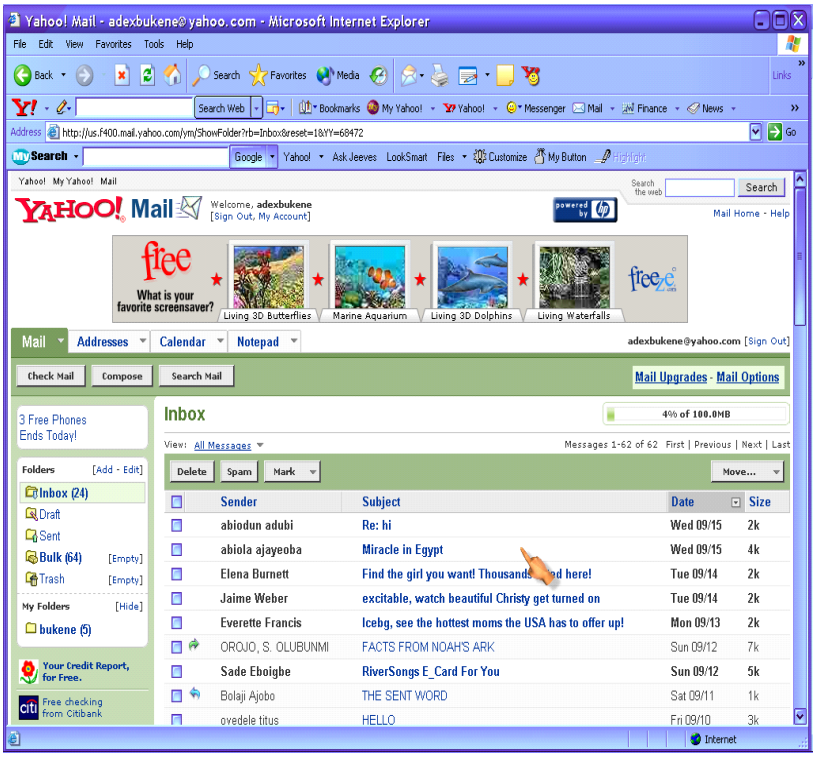


Fig.2: Compose what? A song, a message,...? Do I have to dart about the screen with my mouse, placing it on every icon and menu option and wait for the system to display more information through the pop-up guide? What can be more miserable than groping in the dark?

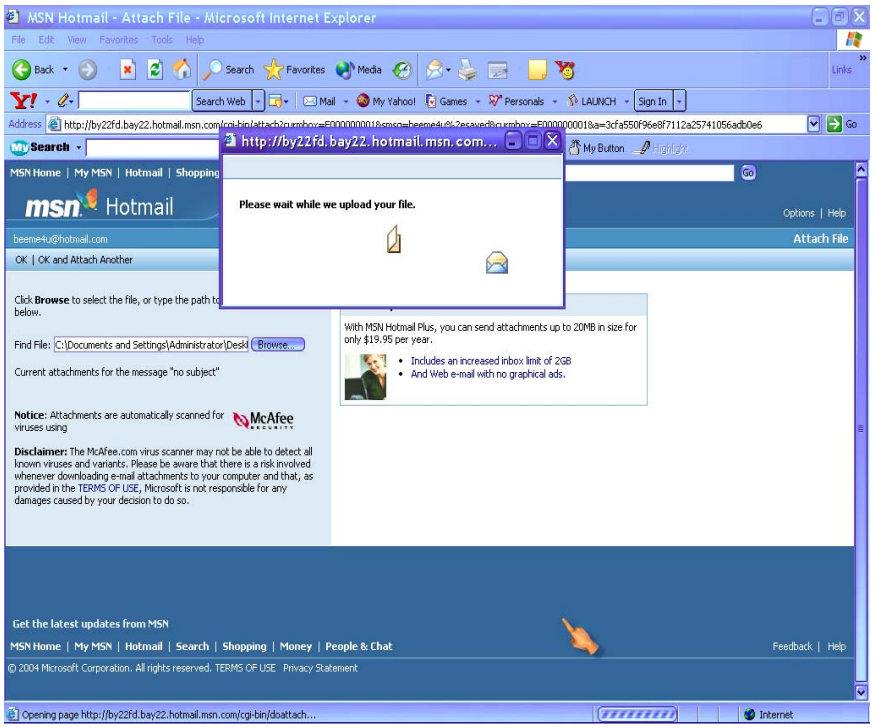


Fig. 3: Though I enjoy the sight of a paper flying from nowhere into an envelope, though my file is a little large (about 0.4 Mb), but it keeps flying as if it wouldn't finish, no measurement to tell when it would be done, has the system hung or frozen? Also, the OK button to start the major operation on this interface is not even remarkably visible, instead, it is hidden in one corner of the screen for only the curiously scanning eyes to locate.

interface factors that are both consciously and unconsciously omitted by system developers.

Implementation

- The Home Page

  The first subject of relevance in implementation is the home page. As it is a general concept that every web application must start with a home called the home page (see fig.4). The page is technically referred to as the index page. For every www.x.com at the URL level, this is the first place a user would be taken to. Of concern now is what makes up the usability of a home page of any website. The answer is not farfetched. It is about a home page that must first *anticipate* the user. That is, it must be able to provide what the user needs. It does not necessarily have to be link to the main issue but a place to meet with the main issue. The main issue as far as this ticketing system is concerned is making reservation i.e. booking a seat for a scheduled trip. On the home page of this ticketing system is this main business straight ahead. Anticipation is one of the key principles of interaction design. This could be explained in the light that not every user is interested in browsing about, or not every one has the patience to do that. This is why what the site principally offers must be presented at the home page. This principle is up to date violated by many web developers because some have been carried away by aesthetics and other kinds of pleasure design. Human Computer Interaction supports aesthetics but should not be traded off for usability goals.

  Also, there is no overload of graphics on the interface as minimalist design is one of the prized principles of interaction design. Colour contrast is observed as not more that three colours were used. There is also colour consistency throughout the pages to be navigated. Input validation is done at client side before submission of entry.

- The Traveller's Sign-in and Registration Pages (see fig.5)

  Another usability feature that is worthy of note is that each page is short and all operations are within the viewable portion of the screen. This is a relevant human computer factor because study and evaluation revealed that some percentage of users does not want to scroll down the screen while browsing the net. So, developer who are completely in the business of designing with "the users in mind" tries to avoid avoidably long page designs.

The error checker of the fields flags for wrongly filled field immediately after filling the field in question. This is more effective than waiting till all the fields are filled. Also, auto-tabbing (a data entry technique which jumps the cursor from one input field to the next) is not supported because it was said that this irritates many users. Some of the error checkers include a check for *incorrect* e-mail address, flagging user for compulsory field that is consciously or unconsciously *omitted*. Also this portion *anticipates* the user by positioning the cursor in the first field (i.e. the username) of the interface. This logically means that the system is ready to accept user data without the need for the user to click the form field as a custom of placing the cursor in

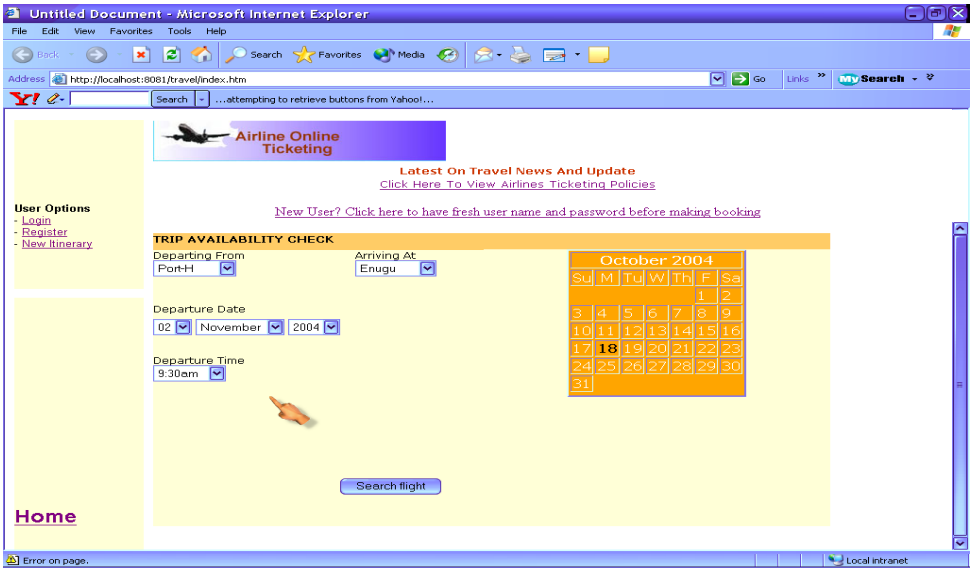


Fig. 4: Shows the home page for ticketing system revealing a home page that anticipates the user. Fits law (www.askTOG.com) defines placing

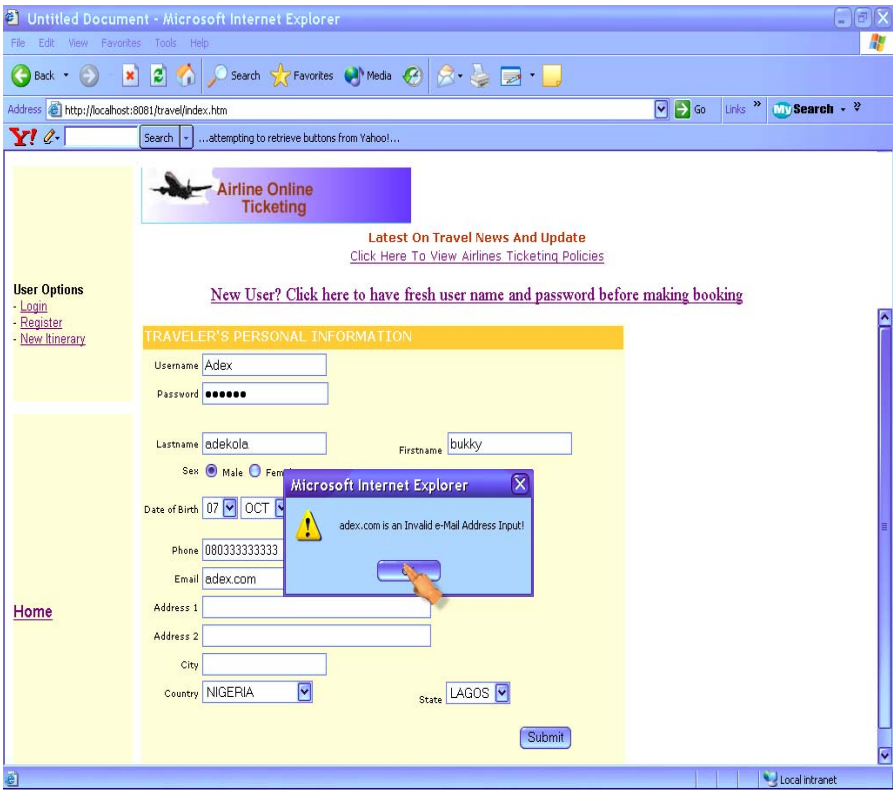


Fig. 5: Shows the travellers sign-in page. This points to the actual error i.e. wrong e-mail input presenting it on the pop-up menu. Worded in language understandable to avoid bugging the user with technical terms.

itfor data entry. Also, input is validated as soon as a user clicks submit button, the relevant omitted field is pointed out. Appropriate feedback is also implemented.

Sets of heuristics chosen for the evaluation of this prototype (the ticketing system)

Ours is a web-based application, so web and some other application heuristics that fit this design were used. It is worthy of note that heuristics for evaluation are decided based on the system/software at hand.

1. Internal consistency: among the consistent things are the page format, links, text colours, text size(as texts of same font but different sizes are completely two different things- Scott W. Ambler (2000) in www.ambysoft.com), page colour etc.
2. Simple dialog: dialog and messages are carefully worded and presented in familiar ways without speaking system-oriented languages. This is demonstrated in the prototype wedesign.
3. Mapping: there is direct mapping of interface objects and action. No hidden function.
4. Visibility: The objects are visible enough on the interface by courtesy of proper colour contrasts. Dark fonts are favoured over light backgrounds.
5. Error prevention: This is implemented for instance where scripting language is used to validate users input as shown above. Error caused due to wrong input is one of the most threatening software errors, which easily give rise to system failures.
6. Feedback: Feedback is demonstrated at every phase of the system use. This prevents user from wandering that can give rise to frustration.
7. Layout: The home page is compact, not crowded with irrelevant things; graphics is conservative (as a principle in HCI says "less is more not more is more"). This aids shorter download time.
8. Fitt's law (www.askTog.com): This states that the time to acquire a target is a function of the distance to and the size of the target. This rule is one of the most ignored principles in design. Fitt's law dictates that the Macintosh pull-down menu acquisition should be five times faster than Windows menu and that is proven out. Fitt's law indicates that the most quickly accessed targets on any screen are the four

corners of the screen because of their pinning action. Also bigger objects are faster accessed. To design for user is also to think about these things. This is shown in the prototype.

9. Anticipation: The user anticipates the user by taking him to the main issue (booking), which also acts as the welcome page and the home.

10. No orphan page: The site purpose is clear at every page as there is a static frame showing the site's purpose on every page. Also each page leads to the "home". No user can get lost. Critical content and navigation options are within the screen view with little or no need to scroll

11. Form fields alignment is appealing and delightful. Wrongly filled ones prompts user at that instance not when all fields are entered.

Concluding Remarks

The interdisciplinary concept of interaction design is fast growing and may soon bring out a standard that will become a defacto standard for software development. This paper focus more on showing interaction design on the interface because user can feel this argument first from the interface. What happens between the user and a system begins at the interface level. That means a poorly designed or overcrowded interface can be off-putting.

This points us to coming up with good designs that users can actually be happy to interact with. Interestingly, lots of so-called interactive software applications, even web-based, are not what they should be but are more or less user-hostile. Evaluation of some web-applications as depicted in this presentation revealed this truth. The problem could be traced to designer's negligence to some interaction design factors. This means, many user-system factors are shunned in the course of their design.

This paper presented the place of human – computer factors, usability and user experience in system designs. A software must be functional but must also be a delight or truly friendly so to speak. Systems must avoid putting frustration to the face of a user. As the interface is the first point of contact, usability and user experience needs to be checked. A system should be good from the interface to the internal functionalities. The search for usability and user experience is what differentiates a human- computer specialist from a software engineer. They need to come together to bring out systems that user really need. To design systems that will put an understandable and controllable face on technology, the heart of a designer at every phase of development should pang "user!, user!!, user!!!....

References

Bruce Tognazzini (2003) . First Principles of Interaction Design. www.askTog.com

Gibbs, W.W. (1994) 'Software Chronic Crisis', Scientific American, 271(3), 72-81

IBM (1993), IBM Dictionary of Computing, McGraw-Hill, New-York.

Norman D.A. (1988) The Psychology of Everyday Things. New York: Basic Books.

Norman D.A. (1999) Interaction Magazine May/June 38 – 42 Affordance, Conventions and Design

Thimbleby, H. (1990) User Interface Design. Harlow, UK: Addison-Wesley

www.ambysoft.com

www.hotmail.com

www.yahoo.com

www.media.wiley.com.software engineering and management practices 4

The Dynamics of Service Differentiation in Computer Networks

Oladeji F. A., Uwadia C. O., Abbas O.

Abstract

The TCP/IP protocol stack controlling the transmission of signals in the Internet provides best-effort service model. In this model, every traffic is treated equally using a single queue buffering and FCFS scheduling. The emerging applications such as the real-time traffic flows require better service principle rather than the best-effort approach. The need to provide differentiated services in order to meet the quality of service requirements of these new applications led to the introduction of Differentiated Service (DS) and Integrated Service (IS) frameworks by the Internet Engineering Task Force. While the IS scalability is not proven, the DS has gained Internet users and researchers' attention. This paper presents the dynamics of achieving service differentiations in a DS-capable network.

1.0 Introduction

The popular Internet is being used for many different activities, including emails, software distribution, video and audio entertainment, e-commerce and real-time games, multimedia etc. Although, some of these applications are designed to be adaptive to available network resources depending on the congestion state along the network paths, they still request for a special quality of service treatment such as low delay, low jitter, no loss or assurance of high bandwidth from the network.

To support this request, there is a need to provide an alternative service model for the Internet, as compared to its conventional one-size-fits-all best-effort service model.

The major problem about the best-effort service model is that it treats all packet flows from different applications on equal basis. A single queue is maintained for all applications and each is attended to on a first come first serve basis. With the emergence of new applications, this sharing approach of the limited resources among users in the network is becoming unrealistic. This is because best-effort system of retransmission and linear increase or decrease (TCP-friendly additive increase, multiplicative decrease may render real time applications useless at the receiving end. A motivation to solve this problem led to the design of the architecture for Integrated Services (IntServ) model proposed by the Internet Engineering Task Force (IETF) in [Braden et al 94]. In this model, all routers along an-end-to-end network path must reserve resources such as bandwidth and buffer for each traffic flow from the same source and maintain a per flow state and per-flow processing at the routers from source to the destination. The complexities of managing infinite flow queues at the router as suggested by this model make the IntServ model difficult to scale well to designers. Also, managing and designing such a model for the present Internet seems non trivial and requires a change to its original infrastructures.

A recent approach to overcome the non-scalability of the IS and foster service differentiation led IETF to propose a new architecture that received a mass attention in Internet community. It is called Differentiated Services architecture (DiffServ) in [Blake et al], its functional component will be shown later.

In this paper, we shall be referring to DiffServ architecture simply as DS and IntServ architecture as IS for easy referencing.

In DS, traffic flows are aggregated and identified as classes. Since the number of traffic classes is expected to be far fewer than the number of flows in IS model, DS is much less susceptible to the scalability problem. The DS service objective is to differentiate among classes of traffic at the routers using a specified per-hop packet forwarding behaviour.

This paper examines existing DS models toward achieving quality of service and service differentiations in computer networks. Section two discusses the current challenges toward computer-based traffic management while section three presents relative DS models being proposed to achieve differentiation according to the type(s) of performance/quality metrics an application

desires. Conclusion is presented in section four.

2.0 Service Differentiation Models

The huge success of the best-effort service model in the Internet is attributed to the underlying transmission mechanism, the Transmission Control Protocol, TCP in [Stevens97]. During the initial stage of the Internet development, most of the applications used TCP as the reliable delivery protocol while its transport layer counterpart, the UDP, was less demanded for because of its unreliability nature. Traffic sources that use the TCP to transmit their data regulate their sending rates (window size) as recommended by TCP/IP linear congestion control measures depending on the congestive state of the network. The TCP transmits data as far as it can meet the requested capacity of the head of queue packet. This type of service is referred to as best-effort service and all traffics are considered equal. The model suites non-real time applications and the operations on them can be resumed any time there is provision e.g. file transfer or email services.

Many revolutionalized applications are emerging from telecommunication industry demanding for a better than best-effort service of this early TCP/IP. Network users place their mission-critical data, day-to-day transactions, web history and conferences including multimedia applications on the public Internet. Such services require strict guarantee on delivery, security and a desired service quality as no delay, low jitter, no loss or high bandwidth from the network. To offer services that would satisfy individual users would require extension to the Internet existing applications. In order to do this, the Internet Service Provider (ISP) has to classify and prioritize traffics according to the application's specific need. In other words, users will be discriminated through delivery of different service qualities such that users who are willing to pay more should receive a better faster service.

These requests for quality assurance, service differentiations and reliability drove the IETF to define two quality of service solutions: the integrated services and differentiated service frameworks. The following subsections discussed the assumptions of these architectures.

2.1 Integrated Service Model

This is the initial proposal for service differentiation in [Bradenetal94]. It seeks to provide strong service guarantee with absolute per-flow bounds on delay, loss rate and throughput. This is achieved by the integration of these four operations: the signaling function, the admission control routine, the packet classification and per flow scheduling policy as shown in Fig 1.

The signaling scheme currently designed is the Resource ReSerVation Protocol (RSVP) in [Bradenetal97]. The applications that require a preferential treatment set up paths and reserve resources in advance using this protocol before actual transmission of data. The admission control routines present at the intermediate routers along the paths determine whether a request for resource could be granted or denied. The classifier

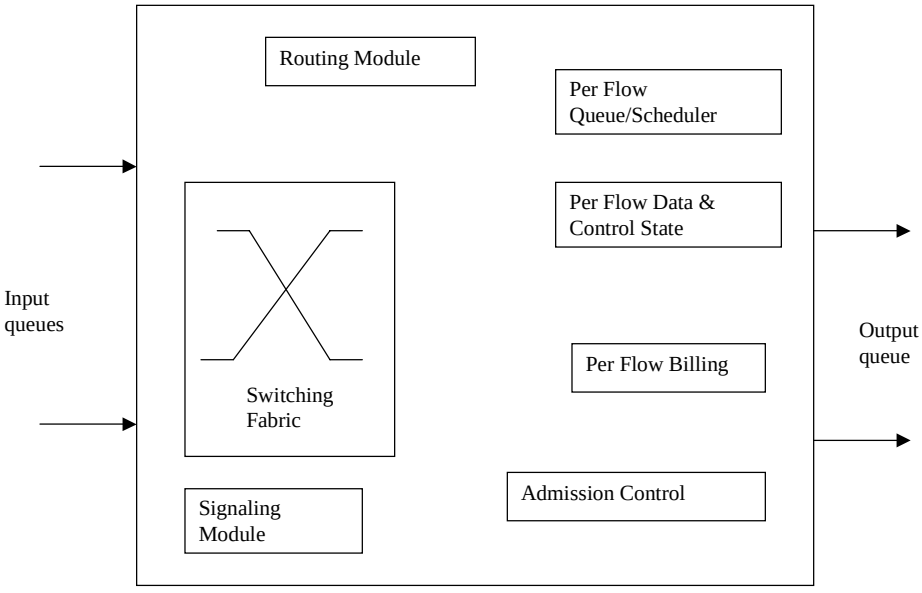


Fig. 1: IS Architectural Layout

program verifies each packet's header and then forwards the packets to the appropriate source-based packet queue. The scheduler then decides which packet to be serviced and ensure its QoS requirements are met.

The architecture as presented in [Bradenetal97] requires substantial modifications to the current Internet infrastructures. This poses concerns on its scalability because maintaining per-flow queue, per-flow scheduling and per-flow billing would impose large computational and memory overheads on the core the routers of the networks.

Despite the flaws in IS, various models such as RSVP, Expected Forwarding (EF), longest source-queue first have been proposed, designed and some are being used in some Internet domains.

2.2 Differentiated Service Model

This model proposed in [Blaketal98] look at ways of delivering differentiated service by (1) defining methods of packet marking with various labels (code points), (2) treating packet differently on a per-hop basis and (3) using service level agreement to give some form of end-to-end performance guarantees across network domains. It proposes the design of traffic shaping algorithms and traffic police routines at the edge routers to enforce quality of service contracts and suggests a form of scheduling scheme to produce the different per-hop behaviors as shown in Fig 2.

The DS model is based on the suggestion that all the traffic flows can be grouped into a finite number of traffic classes. Individual traffic flows with similar Qos requirements are combined together to form traffic aggregates (classes). Each packet class is identified by a short label in the IP header, (Type of service field in [Nicholset98]), called the DS Code Point (DSCP). Whenever a host or edge router sends data into a DS network, it first marks the packet with the appropriate DSCP value. The DSCP value is decided by a complex classification procedure at the edge routers of DS network.

The intermediate routers (hops) treat the received packets based on the associated DSCP value alone. The queue and scheduling behavior to be applied on arriving packet are based on the value of DSCP. The queuing and scheduling behavior in a DS is called the Per-Hop Behaviour (PHB) and it defines how an individual router will treat a packet when forwarding it over to other hops through the network. According to [6], four PHB have been defined and standardized which are: the default PHB, the class-selector PHB, the Expedited Forwarding PHB and the Assured Forwarding PHB.

The default per-hop behaviour, if selected, provides the same kind of service as the existing best-effort model. The class-selector PHB offers seven different queuing behaviors with increasing packet time forwarding probability. The Expedited Forwarding per-hop behaviour offers premium end-to-end service consisting of low loss, low latency, low jitter and assured bandwidth as designed in [Jacobsonetal99]. The Assured Forwarding per-hop behaviour offers different levels of forwarding assurances for the received packets in [Heinanen99]. It delivers IP packets in four

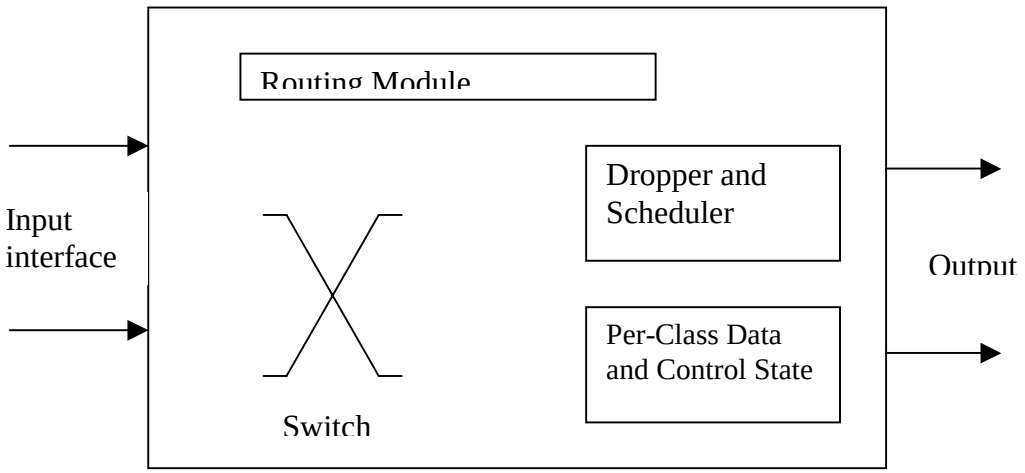


Fig. 2: DS Architectural Model

Table 1: Assured Forwarding Classes and its Recommended Code Points

	Class 1	Class 2	Class 3	Class 4
Low Drop Precedence	AF_{11} (001010)	AF_{21} (010010)	AF_{31} (011010)	AF_{41} (100010)
Medium Drop Precedence	AF_{12} (001100)	AF_{22} (010100)	AF_{32} (011100)	AF_{42} (100100)
High Drop Precedence	AF_{13} (001110)	AF_{23} (010110)	AF_{33} (011110)	AF_{43} (100110)

independently forwarded AF classes (see table 1). Within each AF classes, the IP packet is further assigned to one of the three levels of drop probability or precedence.

DS can be provided either in absolute and relative manner. In the former approach, service differentiation is derived by using admission control, bandwidth broker and resource reservation just as the IS model. The absolute DS aims at achieving performance measures in a way similar to IS model, but without keeping per flow state within routers as specified in [Manimaran02]. It is usually referred to as Virtual Leased Line or Premium DS model. The sources are expected to keep to the traffic profile such as the premium rate to send and are guaranteed the transmission of that rate by reserving resources from the source to the destination of that traffic. It strives to provide strict guarantee service. Violation of the profile may lead to a drop of such traffic flow.

The relative service differentiation, on its own, treats an aggregate traffic class relative to another class. It is achieved through the use of packet schedulers and packet accept/discard rules. In the relative differentiation approach, internet traffic is divided into N traffic classes such that class i gets a better or at least no worse service than i+1 traffic class. This may be achieved through application of priorities (strict, static or dynamic), differential pricing strategy or differential allocation of available resource using different weights with differential penalty during heavy congestion. Providing differentiated services on monetary basis implies that a network user who pays more will expect a lower delay level and loss rate level than a user who pay less. Table 2 compares the IS and DS models.

3.0 Performance Metrics and Differentiation Models

In the DS framework, the quality of service is measured in terms of any of the following performance metrics:

1. Bandwidth – the maximum data transfer rate possible between the source and the destination.
2. Delay or latency – the time a packet takes to traverse from the source to the destination and
3. Reliability or packet loss rate – the average error rate of medium or average number of packet discarded on transit

Most of the congestion responsive-based quality of service models proposed so far aim at controlling one or two of these metrics for a connection session. As would be discussed later, some researchers support building of algorithms that would guarantee each of these parameters independently such that each traffic flow selects the type of quality metrics desired during the session setup. In [8,9], this independent routine was criticized in the sense that it would make traffic management complex at each router end. Moreover, the Internet traffic load fluctuates greatly, thus, any scheme that would regulate and guarantee service performance metrics must be very robust, fair and tolerant to changing network conditions. Schemes that would provide service differentiations using these metrics in fairness to all network traffic class in DS domains are required for deployment on the Internet. Also, any of such schemes must be efficient in managing traffic embedded with the qualities such as fairness, responsiveness, low overhead and social optimality as specified in [Jain90].

Many ISPs researchers are proposing the use of different

Table 2: Comparison of IS and DS Models

Criteria	Integrated Services	Differentiated Services
Granularity of service Differentiation	Per-flow	Per-aggregate
Scheduling and Buffering per router	Per-flow	Per-aggregate
Traffic classification based on::	Several header fields	6bits DS field in IPv4 or IPv6 Header
Admission Control	Required	Not required, but shaping/policing may be required
Signaling Protocol	Required (RSVP)	Not required, may provide semi-static reservations
Type of service differentiation	Deterministic or statistical guarantee	Absolute or relative assurances
Coordination for service differentiation	End-to-end	Local router (e.g. per hop basis)
Scope of Service	Unicast or multicast path	Anywhere in a network or specific paths
Network Management	Similar to circuit-switch networks	Similar to existing P networks
Network Accounting	Per-flow	Per-aggregate
Inter-domain Deployment	Hard (requires multilateral agreements)	Simpler (requires bilateral agreements

techniques to differentiate one application from the other using the theoretical queue models e.g. the waiting time priority in [Kleinrock76]. Differentiation could be viewed in different ways among Internet users. From a performance perspective, a real-time voice or video on demand transmission must experience a lower delay level than an ordinary email or file transfer. In this case, distribution of available bandwidth according to the exigency of traffic is important. From the application's viewpoint, a traffic flow should have access to the available resource since each has its own specified quality of service requirements and thus, fairness becomes an important issue. In case of congestion, to prevent a collapse, unresponsive flows need to be penalized by packet dropping. This may be done to force all traffic to be responsive to the congestion no matter urgent and stringent such traffic demand might be. Currently, most research works focus on metric-based modeling approaches for delivering the Differentiated Services.

3.1 Relative Differentiated Service Models (RDS)

The central idea in relative differentiated service is that the network traffic is grouped into N service class, which are then ordered based on their relative quality:

Class i is better than class i-1, 1<i<N, in terms of local performance measures such as queuing delays and packet losses.

In other words, the class a packet belongs, determines the level of quality it receives from a particular router. This relative differentiated service is supported in [9] with the goal to give a better performance to class i traffic than class i -1 traffic but with a fixed quality of service spacing among classes as a fairness yardstick. If the goal is consistently achieved, then it is concluded that class i users receive better performance than class i–1 users, then, in return, the ISP can legitimately charges class i traffic a higher tariff rate than class i-1 traffic. A popular model for achieving RDS is the proportional model proposed in [Dovrdis00]. Specifically, the model imposes the constraint that the local (per hop) performance measures for each class should be ratioed in proportion to certain differentiation parameters that the network operator selects.

During a connection, the network administrator selects a quality of service parameter to each service class to serve as its Quality Differentiation Parameter (QDP). At a time interval, the average of a selected performance metric is calculated across classes e.g. delay and expressed as a ratio to that of another service class. This is compared relatively to their respective chosen QDPs. If there is a match, the scheduling and

dropping models are assumed to be fair to all classes.

The model has the following objectives:

(i) To provide consistent service differentiation between classes. This is to say that a class with higher advertised quality should consistently out perform a class with lower advertised quality.

(ii) To allow the quality spacing between classes and within classes to be adjusted such that there is fairness in providing the differentiated services.

The two goals must be met at all time scales long or short. Specifically, certain questions sprang up from the proposed proportional RDS model in [Kleinrock76]. Some of them have been analyzed while research is still going on the following:

1. Given desired quality of service request for N traffic classes, under what conditions (e.g. traffic load distribution for N classes) can feasible control parameters be found to achieve the ratios?
2. Given that the ratios are feasible, how can one obtain the control parameters that will achieve the ratios?
3. Given the obtained control parameters, can we maintain the spacing of the control parameters at all time scales?

Some researchers have attempted these questions. Some of their outcomes are presented in the next section. The detail can be found in [Dovrdis00, Dovrdis99, Dovetal02, Leungetal00, Dovrdis02].

3.2 Proportional Relative DS Model

The model states that the network resources should be distributed among users in proportion to the share they have been assigned. So, if q_i, q_j are the network resources being used by user i and user j and s_i, s_j are the network share assigned to user i and user j by the network operator respectively, then, the proportional differentiation model imposes that:

$$\frac{q_i}{q_j} = \frac{s_i}{s_j} \qquad 3.2.1$$

The basic idea is that, even though the actual quality experience by each user will depend on the total load of the network, the quality ratio between users should remain fixed, independent of the load.

In general, suppose that there are N classes and q(I) is an average quality metric of class i in a certain time interval $I = (t_o, t_o + \delta)$ of duration δ, the proportional differentiation model states that for a given monitoring interval time scale δ, the following constraints should hold between the average quality metrics of two classes i and j, for all time intervals of duration δ in which $q_i(I)$ and $q_j(I)$ are defined

$$\frac{q_i(I)}{q_j(I)} = \frac{c_i}{c_j} \qquad 3.2.2$$

The parameters c_i, c_j are referred to as quality differentiation parameters (QDP).

Since higher classes are better in relative differentiation, this QDP should satisfy the relation $c_1 < c_2 < c_3 < \ldots < c_N$

Thus, for queuing delay differentiation, the equation becomes

$$\frac{d_i(I)}{d_j(I)} = \frac{\delta_i}{\delta_j} \qquad 3.2.3$$

where $\delta_1 > \delta_2 > \ldots \delta_N$ δ_i's are the delay differentiation parameters for the respective classes; and for the proportional loss rate differentiation;

$$\frac{l_i(I)}{l_j(I)} = \frac{\sigma_i}{\sigma_j} \quad \sigma_1 > \sigma_2 > \ldots > \sigma_N \qquad 3.2.4$$

Based on this model, efforts have been made to achieve proportional fairness in traffic delay as in [Dovrdisetal99], loss rate as in [Dovetal02], and jitter differentiation in [Leungetal00]. Later in this paper, we shall examine the model using waiting time priority for evaluating delay proportional fairness. Another aspect of proportional differentiation is how to achieve proportional scheduling and dropping of packets when there is congestion. Below, these two issues are examined.

3.3 Scheduling Models in a Proportional RDS Paradigm

In [3], two scheduling algorithms for approximating the proportional differentiated services by adjusting the delay differentiation parameters under heavy load conditions were proposed. The model called Backlog Proportional Rate scheduler (BPR) is based on a general processor sharing model with the modification that the class service rates are dynamically adjusted so that

they are ratioed proportionally to the corresponding ratios of measured class loads.

The second algorithm is the waiting time priority model scheduler earlier set out in [Jain90]. It is based on time dependent priority algorithm and has been the model behind most proportional service differentiations. The priority of a packet in flow i at time t is proportional to the waiting time of the packet at time t, where the proportional constant c_i is a service parameter for class i.

The Time dependent Priority (TDP) is a non-preemptive scheduling algorithm which provides a set of control variables b_i $1 = i = N$ where $0 = b_1 = b_2 = \ldots = b_N$ and b_1 denotes the instantaneous priority of class i packets.

Specifically, if a packet arrives in class i at time T, then its priority at time t (for $t = T$), denoted by $q_i(t)$ is

$$q_i(t) = (t - T)\, b_i \qquad 3.3.1$$

If $N_i(t)$ represents the number of packet in class i, waiting in the queue at time t and if the server is ready to transmit a packet at the head of queue, it will choose a packet from class i* where

$$q_i^*(t) = \max \{ q_i(t) \} \qquad 3.3.2$$

and $N_i(t) > 0$ $\qquad 1 = i = N$

Whenever there is a tie in the highest priority, the tie is resolved using FCFS basis. If there is no packet in the system, then the server will be idle and it will be activated by any newly arriving packet.

The work in [Dovrdisetal99] proposed proportional delay differentiation through the use of packet schedulers. Three schedulers were addressed and their performances compared. They are: the Proportional Average Delay Scheduler (PAD), Waiting Time Priority Scheduler (WTP) and the Hybrid Proportional Delay scheduler (HPD).

The PAD calculates the normalized average queuing delay of all packets serviced from a traffic class in the time interval t and selects the queue with the maximum normalized average delay for scheduling. In this scheduler, normalized average delay is given as:

$$d_i(t) = s_i / \ddot{a}\, p_i \qquad 3.3.3$$

where s_i is the sum of the queuing delays of all the class i packets that have been serviced at the current time t and p_i is the corresponding count of class i packets that have been dequed at the time interval and ä is the delay differentiation parameter of that class.

The WTP attempts to minimize the normalized head of the queue waiting time of different classes. This is expressed as

$$w_i(t) = \frac{w_i(t)}{d_i} \qquad 3.3.4$$

where $w_i(t)$ is the waiting time of the head packet of a class or a queue. The waiting time of a packet is the difference between current system time and the time when the packet is entered the queue.

The WTP schedules packet with maximum waiting time. While the PAD works well in any timescales, WTP only achieved delay differentiation during heavy loads.

The best features of PAD and WTP are combined in hybrid proportional delay differentiation expressed in [13] as:

$$h_i(t) = (g)\, d_i(t) + (1 - g)\, w_i(t) \qquad 3.3.5$$

Where 'g' is the HPD parameter. Under heavy load condition, both PAD and WTP work well but when the utilization decreases under low load, the value of 'g' is set close to 1 so that HPD works more like PAD.

Proportional loss rate differentiation was proposed in [Dovetal02] to schedule packets from queue or class with highest loss rate while in [Dovrdis02], a relative bandwidth differentiation was modeled using WRED and combination of a relative loss and a relative delay differentiations for the TCP flows as:

$$\frac{bw_i}{bw_j} = \frac{\hat{a}_i}{\hat{a}_j} \qquad 3.3.6$$

Where bw_i is the throughput achieved by flow i and $\hat{a}_i$ is the bandwidth differentiation parameter of flow i. This works well in Intserv-based traffic management.

Research efforts are being directed on how to control network flows by a simultaneous control of more than one performance metrics.

A recent proposal for relative proportional DS is the proportional jitter differentiation model proposed and simulated in [14]. According to the model, the ratio of the corresponding jitter differentiation parameters is:.

$$\frac{J_i(t, t+T)}{J_j(t, t+T)} = \frac{c_j}{c_i} \qquad 3.3.7$$

where c_i, c_j are the jilter differentiation parameters and

$J(t, t+T)$. In this model, class i is better than class j if $c_i > c_j$

Thus, $J_1(t, t+T)c_i = J_2(t, t+T)c_2$

$J_2(t, t+T) = J_3(t, t+T)c_3$

$J_N(t, t+T)C_{n-1} = J_N(t, t+T)c_N$

3.3.8

A jitter of a packet k in a class queue i is the difference of queuing delay of this packet and the preceding packet in this class.

$J_i^K = d_i^k - d_i^{k-i+}$

3.3.9

A relative jitter proportional scheduler pseudocode is shown below as used in (Dovetal02):

3.4 Dropping Mechanism

One of the key functions of any congestion avoidance algorithm is to stop admitting newly packets into the queues after a particular threshold is reached, even to drop some of the backlogged packets depending on the level of the congestion. In proportional differentiation, anytime the buffer manager discovers heavy congestion, it invokes a dropper (e.g. RED) to select an already backlogged packet from a target class as follows: The dropper maintains a running average for the performance metric e.g. delay, loss rate or jitter for each class i. When a packet that is backlogged needs to be dropped, the target class to drop from is selected by computing the minimum of their measured performance divided by their quality differentiation parameters e.g. for loss rate, $l_i(I) / ó_i$ is computer for class i at interval I and so on for the other classes. The class with minimum is selected and one or more packets are dropped that would not disturb the proportionality of its scheduler and buffering scheme.

All these models are exposed to criticism before deployment into the Internet. The main yardstick is that, a traffic management algorithm that would be fair, responsive, socially optimal with less overhead are needed high performance of the Internet switching protocols.

4.0 Conclusion

The idea behind service differentiation on networks sprang up as the packet switch-based Internet desires to transmit real time applications to meet current telecommunication demands. Real-time applications that dominate its traffic most essentially require stringent constraints on how and when to forward signals that the best-effort service model of the present Internet could not provide. To accommodate such traffic would require modifications to the current Internet. This trend has led to various propositions from researchers and network designers. Some of their outcomes had been presented in this paper. Efforts are being directed towards IETF Differentiated Service architecture than the IETF Integrated service differentiation framework because of its scalability attribute. Criticisms and evaluations are expected from general public and Internet users before their deployment. Research efforts are still being directed toward managing new applications (unresponsive real time traffic) efficiently with fairness to the existing elastic TCP-friendly applications.

References

Blakeetal98, Blake S., Black D., Carlson M., Wang Z. and Weiss W. (1998) "*An Architecture for Differentiated Services*", Dec. 1998, IETF RFC 2474

Bradenetal94, Braden R. Clark D. and Shenker S. (1994) "*Integrated Services in the Internet Architecture: an Overview*", Jul 1994, RFC 1633

Bradenetal97, Braden R., Zhang L. Berson S. and Jamin S. (1997) "*Resource ReSerVation Protocol (RSVP) – Version 1, Functional Specification*", Sept. 1997, RFC 2205

Dovrdis99, Dovrdis C. and Ramanathan P. (1999) "*A Case for Relative Differentiated Services and Proportional Differentiation*", IEEE Network Oct. 1999

Dovrdis00 Dovrdis C. (2000) "*Proportional Differentiated Services for the Internet*", PhD Technical Report. University of Wisconsin, Madison, May 2000

Dovrdis02 Dovrdis C. and Ramanathan P. (2002) "*Proportional Services Part II: Loss Rate Differentiation and Packet Dropping*", IEEE/IFIP International Workshop on Quality of Service, Jun 2002 pg. 52-61

Dovrdisetal02 Dovrdis C., Stiliadis D and Ramanathan (2002) "*Proportional Differentiated Services: Delay Differentiation and Packet Scheduling*", IEEE/ACM Transactions on Networkiing, 10(1), pp 12-26, 2002

Heinanen99 Heinanen J., Baker F. Weiss W. and Wroclaski (1999) "*Assured Forwarding PHB Group*", Jun 1999, RFC 2597

Jacobsonetal 99 Jacobson V., Nichols K. and Poduri K. (1999) "*An Expected Forwarding Per Hop Behaviour*", Jun 1999, Request For Comments 2598.

.Jain90 Jain R. (1990) "*Congestion Control in Computer Networks: Issues and Trends*", IEEE Network Magazine, May 1990 pg. 24-30

Kleinrock76 Kleinrock K. (1976) "*Queuing Systems Volume II Computer Applications*", Vol. 1, John Wiley, 1976

Leungetal00 Leung K., John C., David K. and Yau Y. (2000) "*Characterization and Performance Evaluation for Proportional Delay Differentiated Services*", Technical Report, Chinese University of Hong Kong, April 2000

Manimaran02 Manimaran S. (2002) "*Scheduling for Proportional Differentiated Services on the Internet*", M. Sc., Electrical and Computer Engineering, University of Mississippi, Dec. 2002

Nicholsetal98 Nichols K., Blake S, Baker F. and Black D. (1998) "*Definition of the Differentiated Services Fled (DS Field) in the Pv4 and IPv6 Headers*", Dec. 1998. IETF RFC 2475

Stevens97 Stevens W. (1997) "*TCP/IP Slow Start, Fast Retransmit; Fast Recovery Algorithms For Congestion Avoidance*", RFC 2001, Jan 1997

Requirements Elicitation as a Success Factor in Software Development

Olugbara, O. O., Ayo, C. K., Adebiyi, A. A., and Akinyemi, I. O.

ABSTRACT

There are many problems often associated with requirements engineering, these include (a) defining the system scope, (b) understanding among the different stakeholders, (c) dealing with the volatile nature of requirements. These problems can lead to poor requirements acquisitions and consequently render the legacy system unsatisfactory or unacceptable. By improving requirements elicitation, the requirements engineering process can be improved, resulting in enhanced requirements and potentially a much better system. Most requirements engineering techniques and tools focus on specification activity that is, the representation of the requirements, which had been found at the long run not to have considered essential needs of different stakeholders. However, the goal of requirements engineering is the production of a good requirements specification. The IEEE 830-1998 Guide defines a good software requirements specification standard. This can only be achieved through proper requirements elicitation practice. This paper focuses on elicitation, in order to address those problems with requirements engineering that are not adequately addressed by several specification techniques and tools. We critically analyze the responses from various software development firms contacted, with a result that little or no attention is given to requirements elicitation, which has often, times leads to software failure. We also describe how elicitation process can be carried out through the use of UML use case diagrams, which is a powerful tool for elicitation.

Keywords: Software Requirements Engineering, Requirements Elicitation, and Use Cases

1 INTRODUCTION

Requirements are capabilities and objectives to which software must conform and are the common thread for all development activities [1]. Software Requirements Engineering (SRE) is concerned with analysis and documentation of software requirements [2]. It is a systematic process of requirements development by iteratively analysing the problem statement and documenting the resulting observations. Requirements engineering is at the heart of system development life cycle with the key objective of specifying a system that at the end will be successful. Requirements analysis therefore, is a process in which what is to be done is elicited and modeled. This process has to deal with different viewpoints, and it uses a combination of methods and tools to produce requirements document.

Requirements engineering is a major problem area in the development of complex software systems [3]. The hardest single part of building a software system is deciding on what to build. Studies have shown that deficient requirements are the main cause of software development failure and incorrect understanding of requirements is one of the most common causes of errors in computer systems. Applications can overshoot their schedule, deliver less than originally promised, or be cancelled before release. Therefore, getting the right requirements is the most important and difficult part of a software development project. Important issues involved in this problem area include [3, 4]:

a. Achieving requirements completeness without unnecessarily constraining system design
b. Analysis and validation difficulty
c. Changing requirements over time
d. Users do not know what is technically feasible
e. Users may change their minds once they see the possibilities more clearly
f. Users may not be able to accurately describe what they do
g. Discoveries made during the later phases may bring about change requirements
h. There are many social, political, legal, financial, and/or psychological factors
i. It is impossible to say with high certainty what the requirements are, and whether they are met, until the system is actually put in place and running correctly

The implication of lack of a complete requirements specification, which evolves from elicitation process, is a grieved one. According to Connolly et al. [5], one of the major reasons for the failure of software projects is lack of complete requirements. The problems that result from inept, inadequate, or inefficient requirements engineering are expensive and plaque most software systems and software development organisations [6]. Despite the current trend to develop applications in Internet time, companies are realizing that the consequences of inadequate requirements engineering are too great [7]. In response to these problems, the software industry is exhibiting an increased interest in requirements engineering and the benefits it yields [8].

Many of the problems in creating and refining a system can be directly traced to elicitation issues. Yet, much of the research works conducted on requirements engineering has ignored this important phase of development. It was asserted that there is a concentration of research in the area of precision, representation, modelling techniques, verification, and proofs as opposed to improving the elicitation of requirements. A conclusion was drawn that research efforts should be directed towards methods and tools needed to improve the requirements analysis process, and in particular, to those providing more support to the elicitation of requirements [3]. The study conducted indicates that requirements errors are passed undetected to the later phases of the software development life cycle, and correcting these errors during or after implementation has been found to be extremely costly. We submit that one way to reduce requirements error is by improving requirements elicitation, an activity often overlooked or partially addressed by requirements engineering techniques.

The goal of this paper is to find out the aspect of software requirements engineering phases, which results into software failure. We conduct an experiment through a questionaire-based findings in an attempt to provide solution. Our opinion differs sharply from those of respondents who believe that software failure is mainly due to programming effort and that lack of skilled programmers is a serious debilitating factor. Report from software houses visited shows that the success of software project rest heavily on ability to program. Stakeholders like programmers and financiers in developing countries like Nigeria believe only on system analysis and direct coding but they are not too familiar with software enginering concepts such as requirements elicitation, specification, verification and management.

This paper highlights the negative consequences of neglecting requirements elicitation and enumerates the potential benefits of its adoption. We argue that software failure is mainly due to lack of requirements elicitation practice, lack of use of development tools and unawareness of software engineering practices. We attempt to fill these holes by exploring requirements elicitation as a success factor.

The rest of the paper is briefly summarized as follows. Section 2 overviews software requirements elicitation. In sections 3, we report our experience and the results of the experiments conducted. Section 4 describes an effective approach for conducting requirements elicitation and we strongly rate use case diagram as a powerful tool for good elicitation practice. To explore the concept further, use case technique is demonstrated with a case study of an aspect of ongoing project at Covenant University [9]. This is done in section 5. In section 6, we describe modeling support tools for practicing requirements engineering with strong emphasis on C# Builder Architect, a new commercial product from Borland and we draw conclusion in section 7.

2 SOFTWARE REQUIREMENTS ELICITATION

Software requirements elicitation is a process by which all parties involved in the development of a software system discover, review and understand user needs and the limitations of the development activity and the software [2]. Requirements elicitation serves as a front end to systems development, which involves various stakeholders: analysts, sponsors, funders, developers, and end users. Requirements engineering process can be broken down into three main activities [3]:

a. Generate requirements from various stakeholders but users should be the major focus
b. Ensure that the needs of all stakeholders are consistent and feasible
c. Validate the requirements derived in line with the needs of the stakeholders

This model involves a sequential ordering of activities in iterative mode and the activities should be properly documented. Requirements elicitation can be broken down into the activities of fact-finding, information gathering, information documentation and integration. The resulting product from elicitation is a subset of the goals of the various parties, which describe a number of possible solutions. The remainder of the requirements engineering process concerns the validation of this subset to see if it is what the stakeholders actually intended. This validation typically includes the creation of

models to foster understanding between the parties involved in requirements development. The result of a successful requirements engineering process is a requirements specification, where the goodness or badness of a specification can be judged only relative to the user's goals and the resources available [3].

A good requirements elicitation process supports the development of a specification with a set of attributes such as those define in [3]. The problems with requirements elicitation inhibit the definition of requirements, which are unambiguous, complete, verifiable, consistent, modifiable, traceable, usable, and necessary. However, the processes of fact-finding, information gathering, information documentation and integration will be refined to specifically address the problems encountered during requirements elicitation. This is where our own contribution comes into play. Problems of requirements elicitation can be summarized into three major categories [3]:

a. Problems of scope, in which the requirements may address too little or too much information
b. Problems of understanding, within stakeholders such as users and developers
c. Problems of volatility, which is the changing nature of requirements.

The following is a set of recommended requirements elicitation techniques [10]: Interviews, Document analysis, Brainstorming, Requirements workshop, Prototyping, Use cases, Storyboards and Interface analysis. These techniques can be used in combination and they are effective in emerging the real requirements for planned development efforts and models.

3. EXPERIMENTAL REPORT

Primary data was collected from programmers and analysts of software development firms through structured questionnaire that was administered unto them. In all, a total of 128 people were administered the questionnaires for the purpose of gathering data, of which 100 people fully responded to the questionnaire, and their responses was used to infer whether requirements elicitation is of paramount importance in software development or not. Thereafter, statistical analysis was carried out on the primary data gathered with sole objective to examine the impact of requirements elicitation in overall software development process. The finding shows that requirements elicitation is significantly important in the process of software development which is often neglected. Chi–square was used to test the following hypotheses.

H_0: Requirements elicitation is significant in software development

H_1: Requirements elicitation is not significant in software development

Testing the hypotheses at 0.05 level of significance, the null hypothesis was accepted, showing that requirements elicitation is of significant important in software development contrary to the report in Fig. 6. The various charts below show the responses from the respondents for which questionaires was administered.

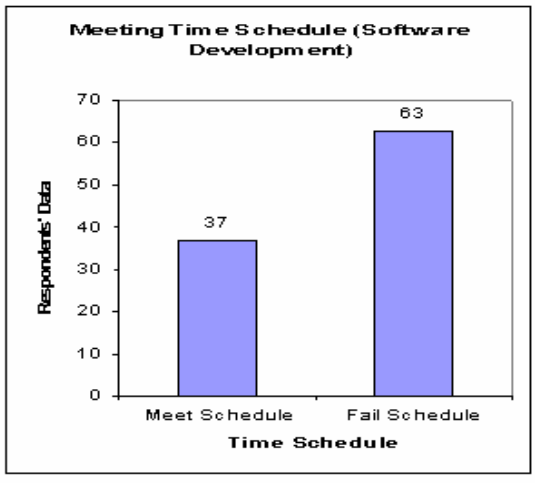


Fig. 1: Meeting Time Schedule

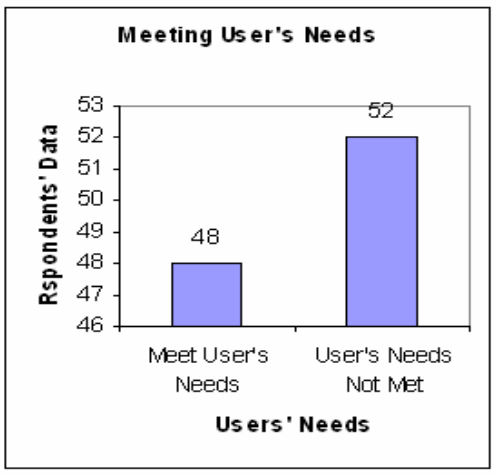


Fig. 2: Meeting User's Need

From Fig.1, 63% of the respondents agreed that they failed to meet the time schedule for software development of their clients due to constant changes in requirements. This inferred that improper requirements elicitation was set before development effort began.

From Fig.2, 52% of respondents attested to the fact that not all their software development efforts meet user's needs as a result of improper communication between the system analyst and users. They identified knowing real customers need is a major task. They acclaimed this problem of not meeting user's needs to not identifying and involving all the stakeholders in their software development efforts.

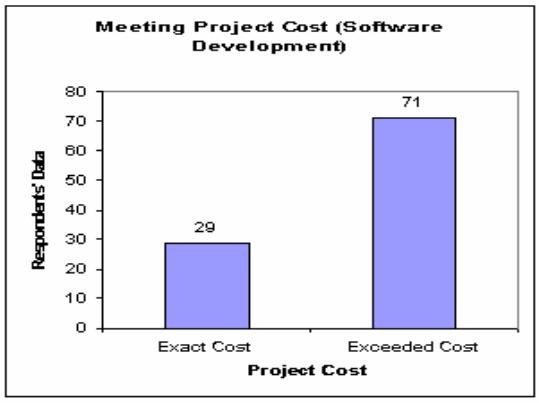


Fig. 3: Project Cost in Software Development

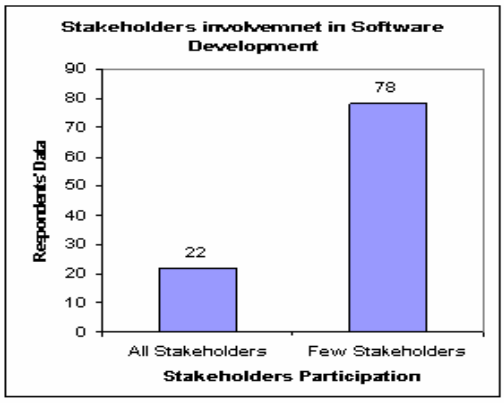


Fig 4: Participation of Stakeholders

From Fig.3, 71% of the respondents agreed that they exceeded project cost due to not meeting user's needs at the schedule time and in most cases it is as a result of poor requirements elicitation.

Moreover, it is evident from Fig.4 that stakeholders are not totally involved in software development, which eventually caused the cancellation of the project or fail to meet users specific needs. The problem is as result of lack of proper elicitation.

Fig. 5: Problems with Stakeholders

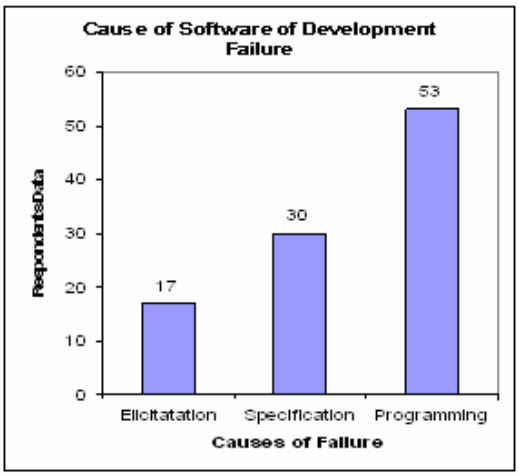


Fig. 6: Cause of Software Development Failure

From Fig.5, 56% of respondents confirmed that they do have problems with various stakeholders in their developmental effort. These problems range from users inability to express their needs to the analyst, lack of team spirit and commitment to the project.

Finally, from Fig.6, 17% of respondents attributed the cause of sofware devewlopment failure as result of requirements elicitation, 30% said it is caused by specification, while 53% said it is caused by poor programming effort. We gather that the respondents believe in software implementation than any process in sofware development phase. However, by our

own understanding and knowledge, requirements elicitation is more crucial to the success of software development than software implementation especially now that code generator tools have been successfully developed.

4. CONDUCTING REQUIREMENTS ELICITATION

The following approach is recommended for carrying-out requirements elicitation. The approach is based on extensive review of literature combined with practical experiences of requirements analysis. The purpose is to solve the requiremens elicitation problems earlier identified,

a. Write a project vision and scope document
b. List a project glossary and acronyms that provide definitions of domain words that are acceptable to the stakeholders for the purpose of effective communication
c. Evolve the user requirements by focusing on product benefits and general acceptability
d. Document the rationale for each requirement when necessary
e. Provide training for requirements analysts and user representatives with the following objectives:
    i. To identify the role of the requirements analyst, who are to work with end-users to evolve the requiremnts
    ii. To write good requirements document
    iii. To identify requirements errors and corrective measures
    iv. To identify investment needs on the project
    v. To understand the project and/or organization's requirements process
    vi. To suggest methods and techniques that will be used for the elicitation
    vii. To understand the use of project's automated requirements tools
f. Prioritize the requirements appropriately
g. Establish a mechanism for requirements management
h. Use peer reviews and inspections of all requirements work products
i. Use an industry-strength automated requirements tools
    i. Assign attributes to each requirement
    ii. Provide traceability
j. Use hybrid of good requirements gathering techniques such as requirements workshops, prototyping, and use cases
k. Provide members of the project team (or stakeholders) with audience–specific versions of the requirements when information is being shared
l. Establish a continuous improvement ethic, teamwork approach, and a quality culture
m. Involve customers and users throughout the developoment effort
n. Perform requirements validation and verification activities in the requirements gathering process to ensure that each requirement is testable
o. Store requirements in a requirements repository instead of a paper document
p. Keep the granularity of the requirements repository small so that individual requirements can easily be entered, iterated, approved, traced, managed and published
q. Ensure that requirements repository stores all kinds of individual requirements metadata and requirements models

5. USE CASES TECHNIQUE FOR EFFECTIVE REQUIREMENTS ENGINEERING

Use cases are pictures of actions a system performs, depicting the actors and they describe the behaviour of a system when a particular stimulus is sent by one of its actors. Use cases are used during the analysis phase of a project to identify and partition system functionalities. They are powerful tools when combined with their textual description to give better semantic information. Many developers believe that use cases and scenarios facilitate team communication. They provide a context for the requirements by expressing sequences of events and a common language for end users and the

technical team. Use case modelling facilitates and encourages stakeholders' participation, which is one of the primary factors for ensuring project success. In addition, it provides a means of identifying, tracking, controlling and managing system development activities. Several other benefits of use cases are reported in [11]. Other requirements elicitation techniques should also be used in conjunction with use cases to provide enough information that enable development activities.

There are two primary artifacts involved when performing use-case modeling. The first is the use-case diagram, which graphically depicts the system as a collection of use cases (represented by oval shape), actors (represented by stick object) and their relationships, which are basically: association, extension, dependency and uses. Use case and actor icons are assembled into large system boundary diagram. This diagram shows all use cases in a system surrounded by a rectangle. Outside the rectangle are all actors of the system and they are tied to their use cases with lines. Inside the system boundary are use cases that are part of the system being modelled and everything outside is external to the system. The diagram communicates the scope of business events that must be processed by the system. The details of each business event and how

the users interact with the system are described in the second artifact, called the use-case narrative, which is the textual description of the business event and how the users will interact with the system to accomplish the task. For detail description of use cases we recommend the materials [11, 12]. Below is a simple example of uses case narrative and use case diagram of a case study of digital Course Registration System (CRS) [9].

Use Case 1	**Add and Drop Course**		
Goal in content	Student could be required to add or drop a course via CRS if registration process is opened, a user interface is displaced with add and drop buttons activated		
Level	This is an extend use case from Add course use case		
Parameters	In: matricNo, course code, college and department hosting the course. Out: current session, minimum and maximum unit allowed per semester, unit registered per semester, students' name and photograph, list of courses available in the department and their corresponding prerequisites		
Preconditions	Course registration is not closed; user is a good academic standing student of the university. If user is a current student, course prerequisite is not checked. Graded course cannot be added or dropped. Course unit registered per semester cannot exceed the maximum or fall below minimum allowed		
Post-conditions (Success End)	Adding or dropping of course(s) is achieved		
Post-condition (Failed End)	Adding and dropping of course(s) is not achieved, or cancel button is pressed and then the process is cancel		
Actors	Students		
Trigger	A student requests to operate the CRS for a CRS operation such as Add or Drop		
Description (Event Flow)	Actor action	System respond	Affected data objects with operations
	1. Student requests to operate the CRS by clicking the correct button	2. Verify registration Status	Reading from the system data file for current registration status
		3. Displays rich interactive user interface	Read and display the lists of colleges and departments available, maximum and minimum units registrable, and current session

	4. Select course hosting college and department from appropriate lists	5. Displays courses and their prerequisites	Read and display the lists of courses and their prerequisites
	6. Enter MatricNo and press start button	7. Verify student	Read from student biodata file and display courses registered
	8. Select a course from course list and press Add button	9. Verify course to add	Update course unit registered per semester and add course to course registration list
	10 Select a course from course registration list and press Drop button	11. Verify course to drop	Update course unit registered per semester and drop course if possible
Extensions			

Table 1. Use Case Narrative for CRS

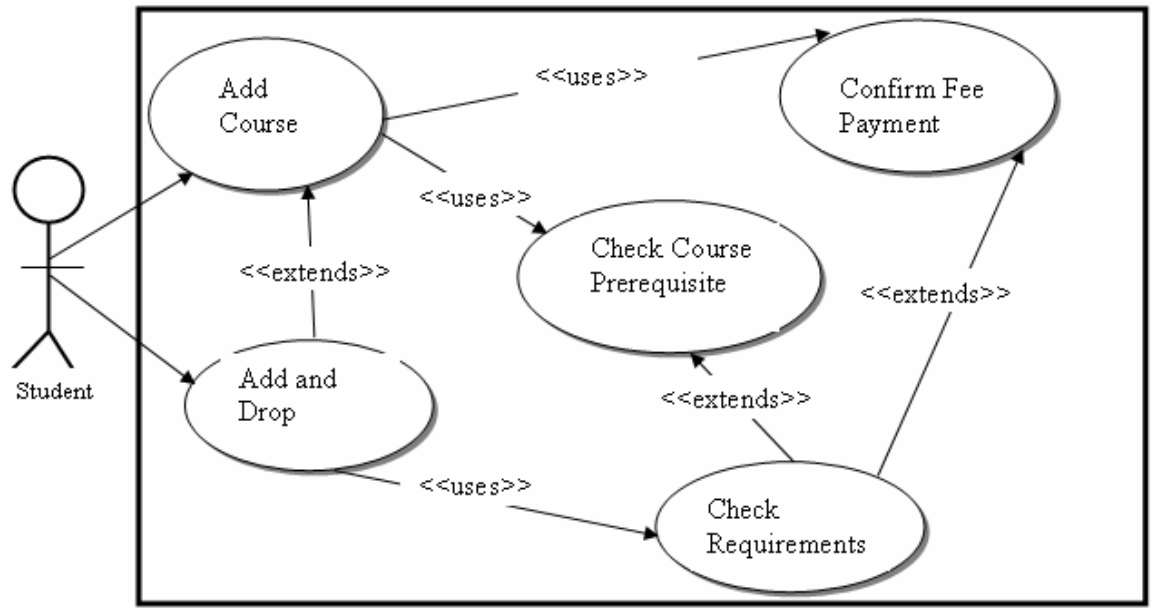


Fig. 7: Use Case Diagram for CRS

6. MODELING SUPPORT TOOLS

In recent years, legacy applications have steadily grown larger and more complex. Monolithic systems have given way to multi-tier applications and software are now embedded with conceivable type of hardware: phones, automobile, ovens and automated teller machines. These essential requirements have made engineering task more arduous and challenging. The appearance of more powerful and user-friendly requirements tools, will definitely minimize this enormous task. Modern integrated requirements tools must support all requirements engineering tasks such as: elicitation, specification, management and other related task outside requirements engineering activity such as: scope management, version control, configuration control and quality assurance or control.

The most related work in this direction is due to Firesmith [13]. The author addressed several problems with traditional paper-based requirements specification approaches and recommended a requirements specification approach to solve these problems that is founded on the use of modern requirement tools based on fine–grained requirements repositories. Using an appropriate tool based on these recommendations, one can automatically, easily and inexpensively generate

various types of high-quality requirements specification that are tailored to meet the individual needs of their various audiences. Firesmith proposal is believed to go beyond the simple use of international (e.g. IEEE830-1998), industry (e.g. OPEN or RUP) or business internal templates for one or more requirements specifications. It also includes the creation of arbitrary requirements reports for requirement management and other purposes.

The major difference between his work and ours is that while the former tends to focus more on requirements specification, the latter identifies requirements elicitation as the pivot or source of most errors that hinders the success of software development. We believe that any improvement at this stage will greatly alleviate the difficulties. On this basis, we view software development stages as illustrated in Figure 8. By refinement we mean transformation of one source to the other. Therefore, specification refinement is the process of going from a specification to an implementation.

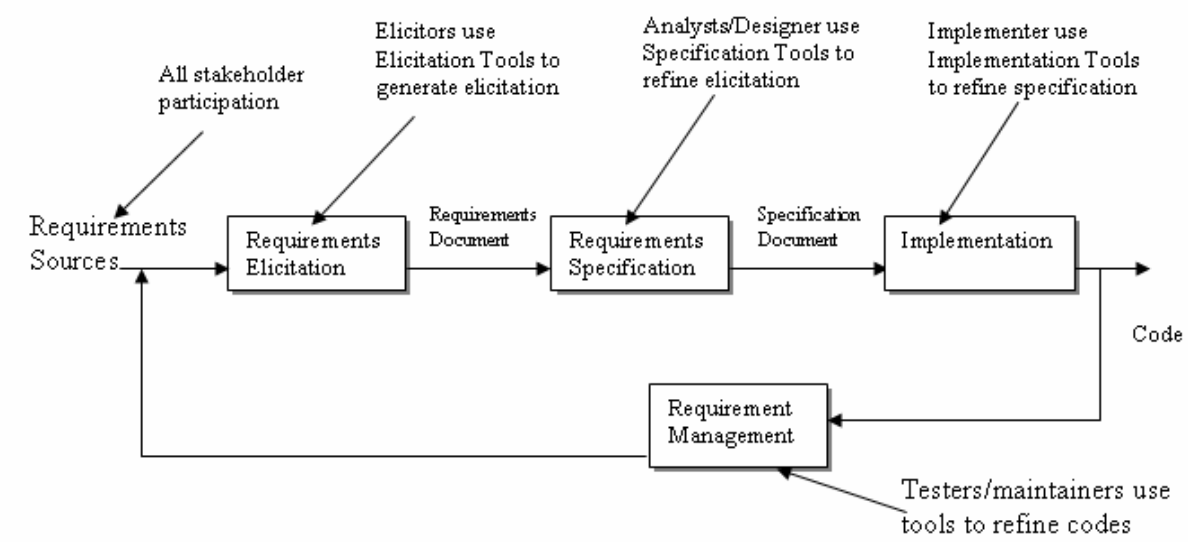


Fig 8: Models of Software Development

A commercial product we are currently investingating to practice this model (Fig 8) is the Borland C# Builder Architect [14]. Borland C# Builder brings the object Management Group (OMG) Model-Driven Architecture (MDA) initiative to the market with one of the tightest integrations of a GUI builder, a modeling tool and a UML execution run-time platform for the Microsoft .NET framework. The Borland product has succeeded in bringing us to higher levels of productivity through a unique approach, which combines a two-way UML/C# synchronization engine that can generate minimal amount of code necessary to unleach strongly typed OOP.

C# Builder Architect auguments the feature set of traditional .NET code, in a pre-built services named Borland Enterprise Core Object (ECO). These services ensure industrial strength handling of associations, derived attributes, persistence, subscription, region-based optimistic locking, object versioning, OCL-based constraint checking and query capabilities, and much more [14]. ECO is a set of software components designed to involve the use of UML at initial design, execution and design evolution (refactoring and extension) times. The largest ECO components are those that act as a run-time environment for executing a UML model and the design time ECO components include UML/C# snychronizer (code generator), handle components to visually connect. ECO objects to UI elements and model validation. The ECO objects extend .NET objects and the vision goes beyond .NET initiatives since .NET objects are not UML objects and the .NET CLR is not as rich as UML models. ECO objects are a particular implementation of UML objects and as such provide the extra services above that of .NET objects.

7. **CONCLUSION**

Software requirements engineering has a goal of specifying a succesful software product. The main reason software projects fail is lack of complete requirements specification, which often results from lack of elicitation practice. The problems created by inefficient requirements practices are expensive and plaque most software development efforts.

These problems are best overcome, by the application of established best practices of which elicitation should be given high priority. Early fault detection in the software development process can significantly cut cost and more importantly, with improved technology for automatic specification generation from requirements and automatic code generation from specifications, it is possible to have a highly reliable system design right from the start of the project.

REFERENCES

[1] Davis, A., and Leffingwell, D. (1999), Making Requirements Management Work for You. http://www.stsc.hill.af.mil/crossTalk/1999/Apr/davies.asp

[2] Thayer, R. and Dorfman, M. (1999), Software Requirements Engineering, 2nd Ed. IEEE Computer Society Press

[3] Micheal, G. Christel and Kyo, C. Kang. "Issues in Requirements Elicitation". http://www.sei.cmu.edu./pub/documents/92/reports/pdf/tr12.92.pdf

[4] Anthony, Aby (2000): Requirements Engineering. http://www.opencontent.org/openpub

[5] Cannolly, T., Begg, C and Strachan, A. (1999), Database System: A Practical Approach to Design, Implementation and Management, 2nd Ed., Reading, M.A. Addison-Wesley.

[6] Sawyer, P., Sommerville, I., and Viller, S. (1999), Capturing the Benefit of Requirements Engineering, IEEE Software, 16(2), pp 78-85.

[7] Weigers, K. (2000), When Telepathy Won't Do: Requirements Engineering Key Practices, http://www.processimpact.com/articles/telepathy.html

[8] Wolak, R.G. (2001), DISS 725-System Development: Research Paper 2, Software Requirements Engineering Best Practices, School of Computer and Information Sciences, Nova Southeastern University

[9] Ibiyemi, T.S., Olugbara, O.O., Ikhu-Omoregbe, N.A. and Osamor, V. (2003), Developing Web-Based Enterprise Application for Effective College Administration, CST Project, Covenant University

[10] Ralph R. Young (2002): "Recommended Requirements Gathering Practices" htttp://www.stsc.hill.af.mil/crosstalk/apr2002.htm

[11] Whitten, J.L., Bentley, L.D. and Dittman, K.C. (2004) Systems Analysis and Design Methods, New York, McGrawHill, 6th Edition

[12] Xiaoqing B., Peng, L.C., Huaizhong L (2004): An Approach to Generate the Thin-Threads from the UML Diagrams. COMPSAC pp. 546-552

[13] Firesmith, D (2003), Modern Requirements Specication, In Journal of Object Technology, Vol.2, No.1, pp. 53-64.

[14] Floury, C, (2003), Unleaching Enterprise Models with Borland C# Builder Architect. A Borland White Paper

TRADRAD - A Hybrid Software Development Methodology

Angela Makolo

ABSTRACT

The choice of software development methodology to adopt when designing an application continues to be a source of concern to programmers in particular and IT professionals in general. Software development methodology plays a prominent role in effective software development process. This paper presents a new development methodology – TRADRAD a hybrid of the traditional waterfall model and Rapid Application Development (RAD) model. It examines its unique features which makes it a useful model in software development with a view to improving productivity.

1.0 Introduction.

Several software methodologies are available with their characteristic features and peculiarities. Some of the methodologies are: the traditional waterfall model, the Rapid Application Development, Spiral Model, Prototyping Model, Fountain Model, Incremental Model and Build and Fix Model. TRADRAD A hybrid of Traditional waterfall and Rapid Application Development software development methodology, which when adopted in the design of an application, results in improved productivity, both of the programmer and the organization as a whole.

The problem of improving productivity remains a major concern in any organization. Productivity simply put, is the efficiency with which output is produced by a given set of inputs. It is generally measured by the ratio of output to input. An increase in the ratio indicates an increase in productivity. Conversely, a decrease in the output/input ratio indicates a decline in productivity. In software development, productivity is the ability to create quality software products in a limited period with limited resources. Quality being measured in terms of the application being able to solve the client's problems effectively and consequently reduces overhead cost of the organization.

The measurement of productivity of programmers poses a lot of problem because of the difficulty of getting standard and effective metrics.

Various metrics have been identified for measuring programmers productivity such as number of lines of code, function points, object point etc. In this work, our focus on productivity is in terms of development time, end users satisfiability, ease of maintenance especially in terms of bug detection and code documentation, change management issues and quality control.

TRADRAD combines the advantages of development using the sequential traditional waterfall, its total managerial control with the rapid application development approach of RAD while emphasizing on regular Joint Application Development (JAD) sessions with the end users.

2.0 LITERATURE REVIEW

Programmer's productivity has been studied for several decades, it has been determined that in general, it can be measured, evaluated and improved. Below is a brief literature review of some studies and researches carried out in evaluating the programmer's performance with a view to improving his productivity.

2.1 Programmers Productivity Metrics

Different software analysts in evaluating programmer's productivity had used various productivity metrics.

Livermore [4] in his study titled "Measuring programmer's productivity" identified some of these metrics. He said, "It should be possible to identify the most commonly used productivity metrics and also the problems associated with these metrics. Research should enable an organization to establish a metric program that will allow for the proper evaluation of the people, tools, and techniques used in the software development process"

He defined Productivity metrics as the ratio of units of output divided by units of input. The productivity ratio

will vary widely from programmer to programmer. Some programmers are capable of turning out ten times the amount of source code than others. The most common measure of unit of output is the number of lines of code (LOC). It is a simple process to count the number of lines of code in a program and there are many automated tools that can be used to do this. If an organization chooses to use LOC as one of their metrics, consistency is vital. There are many different definitions of what a line of code is. If one programmer counts comments, blank lines, and data definitions their LOC output will be much greater than a programmer who is only counting executable lines.

There are several drawbacks to using LOC as a productivity metric. LOC can only be measured after the software is completed. It is difficult to estimate LOC and use interim LOC counts to measure progress. The number of LOC used to implement functionality varies greatly between programming languages and prevents direct comparisons of programmer productivity if different programming languages are being used.

An alternative to measuring lines of code is to measure function points with a technique called Function Point Analysis (FPA). FPA was developed in the 1970's to measure deliverables instead of lines of code. As a metric, function points represent the components seen by the end user including: inputs, outputs, inquiries, external interfaces and internal files. There is only a very loose correlation between function points and LOC .Measuring function points can be very difficult and it is easy to under count the number of functions. Function points need to be determined by trained specialists and only large organizations can afford the luxury of maintaining a staff of FPA specialists. FPA does not work well in certain types of applications.

Clearsmith [2] in his article "Evaluating Productivity" highlights the problem of getting a standard productivity metrics to evaluate programmer's productivity.

He states, "If the goal of a software-development project is to have working, maintainable software, why do so many managers focus on hourly rate, rather than some more direct measure of productivity? Might it be that they lack a more direct measure of productivity?" He compared programming as an act of creation to writing expository prose [2].

Donald Knuth [3] wrote in *Literate Programming,* "The practitioner of literate programming can be regarded as an essayist, whose main concern is with exposition and excellence of style. Programmer productivity and skill levels vary by orders of magnitude, but this variation is not something found in a resume. Essayists live in a relative meritocracy, where editors can read and assess their work; programmers are generally hired and managed without reference to their past work, which is generally protected as secret information owned by the previous employers. Even if the programmer can show software that they've written on his or her own (their own copyright, or open sourced), a manager who has not programmed in the last handful of years is likely ill-equipped to evaluate the code's quality. Moreover, the source code does not tell the interviewer:

- Who really wrote the code?
- How long did it take this person to write?
- How unique and difficult was the initial programming task, as posed to the programmer?

The situation is comparable to having a completely illiterate book publisher.

For decades, there has been in academia a small industry of research into programming-productivity metrics. Starting in the 1960s, the earliest metric, measurable quantity for assessing programmer productivity was lines of code. Like hours clocked at the office, this measurement is simple for a manager to make. Unfortunately, the worst programmer in the world can (and will, given the incentive of the lines-of-code evaluation) generate reams of heavily commented, sparsely formatted, inappropriate, malfunctioning, and unmaintainable source code. Judging current and potential employees on dollars-per-line is route to bug-ridden mayhem".[3]

Other metrics, slightly more difficult to obtain, involve counting reported defects and specified features-perhaps from bug-tracking and requirements databases. A few companies employ, grudgingly, a smattering of these software metrics to conform to customers' notions of quality control. There are scores of such metrics, but none of them appear in résumés, and managers rarely use them to evaluate employees, teams, or projects. This neglect of numeric measures is probably a good thing; designing and writing code is *not* an item-per-hour task instead, the proper evaluation of a programmer requires a skilled reader of source code. [3].

2.2 Improving Productivity of Software Development

Improving productivity is a major issue in software development, Alan Rouhana [1] in his research work titled "Improving speed and productivity of Software Development" based his study on the survey of software management practices in western Europe, Japan and the United states. He studied practices that accelerated

development time of a software project and increase its productivity, he identified time as a fundamental measure of performance in software development, since the result of a late project could be quite devastating and embarrassing, for instance, customers will not wait for a delayed product when alternatives are readily available and if the software misses the launch date the product enters the market without the features which might further differentiate that product in the market place. However project delays create inescapable deadline pressure and a defective product will be delivered to the customer and he would unlikely purchase future released products.

He highlighted the fact that development speed and productivity are asymmetrical because a low productivity organization can be quicker to market a product by throwing more human resources, keeping in mind that by adding human resources at the latter stages of the process will extend development time.

He also did a survey, which entails a Global Comparison of Software Management Practices. It showed that all the firms in the different regions Japan, United States and Western Europe allocate the effort in term of man-months per stage. In the same proportions while looking at the time cycle between the regions, all the firms assign 15% of the time to the specification phase and the difference was between Japan and Europe in the coding and implementation phases; the Japanese spend more time in implementing and less time in testing (average 22%) while the European spend more time in testing (average 29%) so the time allocated in average is higher.

To learn about the importance of certain project management factors, the survey also asked the respondents to identify factors that helped reduce overall software development time. These are:

- The use of prototyping to demonstrate how the proposed software will work;
- Better customer specifications initially;
- The use of CASE tools and technology;
- Concurrent development of stages or modules;
- Less rework or recoding;
- Improved project team management;
- Better testing strategies;
- Reuse of code or modules;
- Changes in module size and/or linkages (smaller modules and standard interfaces speed, coding and testing with parallel development);
- Improvements in communication between team members; and
- Better programmers or software engineers.

He conducted a survey to determine if significant differences existing in software management by these regions (western Europe, Japan and US) understand the best practice and where it is conducted. In the survey, he asked firms to describe a recently completed software, the type of project, language used, procedures to manage and monitor the process, performance measures and some quantitative information like the project size, productivity over time, allocation of time, effort and team size among project phases, the degree of newness of the project, the effectiveness of different tools and techniques for time compressing the development process and the stages of the process in which the reductions have been achieved.[1].

3.0 FEATURES OF TRADRAD

TRADRAD methodology is a good tool to adopt in implementing any project (Irrespective of its size and complexity) when the development team is a small group of highly skilled professionals, committed to timely product delivery, with easy access to end users of the application and when they have a proper understanding of the manual or electronic system already in place in the organization.

TRADRAD compresses system development life cycles resulting in appreciable reduction in development time.

It advocates total involvement of the main stake holders of the application in all the development stages through the regular joint application development (JAD) sessions with the developers. This ensures that a thorough understanding of the system by both parties is easily attained. The developer's learning curve is reduced since he can easily clarify any ambiguous specification.

It also supports the use of structured programming techniques, which incorporates the advantages of modular programming into the methodology. This includes easy code reuse, ease of documentation and maintenance, modularization etc.

4.0 TRADRAD METHODOLOGY

The stages of system development using TRADRAD methodology includes information gathering, system implementation, documentation and maintenance.

Information Gathering: The stage of gathering information about the system to be designed, involves

problem definition and setting the requirement specification.

This stage of development is key to success since a mistake in defining the problem will invariably lead to unsuccessful implementation.

To define the problem, a joint application Development session would be organized involving the programmer, who could double in as system analyst and the key users of the system..The full involvement of the end users in every stage of development ensures that any mistaken specification is easily detected and corrected . The result of this is a detailed system requirement / specification document which will form the basis of the development work.

Here, the main functionalities of the system are defined and prioritized, the estimated development time is determined and the frequency of the JAD session is set. The functional specification of the system are designed and approved.

System Implementation: This stage combines the system design, data modeling, coding, testing and debugging. After the first JAD session, the programming team, has a thorough knowledge of the system, the next step is to analyse the system requirement and modularize it into smaller units, then decide how feasible the requirement is in terms of its being programmable and determine the best programming tool to use; TRADRAD supports object orientation and modularization. A prototype of the system is developed, with little emphasis on functionalities, another JAD session is organized to demonstrate the prototype to the users, after approval, the actual coding begins.

The decision on the programming language to use, the design methodology, data structures and schemas are made here.

Depending on the size of the programming team, the project leader assigns various modules of the application to the various programmers and a deadline is fixed so that all the modules can be knit together in preparation for the next JAD session with the end users.

Usually, the JAD session is organized to demonstrate an already implemented part of the requirement, which is then tested by the end users and debugging by the programmers take place simultaneously.

This methodology combined the systematic approach of the traditional waterfall with the compressing feature of Rapid Application Development, that is its ability to compress the analysis, design, coding and testing into a series of iterative development cycles.

Each iteration delivers a functional version of the final system and can be modified easily by the programmer once the need is identified in the JAD session, this adds the advantage of ease of change management to TRADRAD.

The choice of programming language is usually influenced by the size of the application, the level of complexity (i.e it is data intensive application or a logic driven one) the expertise and skills of the programming team .

From experience, end users always want a complete automated process which can be initiated just by clicking a button, thus a web enabled windows application is always preferred.

The implementation is completed after ascertaining that the system meets user specification and a final signoff is obtained.

Documentation: This task is very important, since any application cannot be deemed complete until the end users can use it effectively. This involves writing user documentation or online help files and a detailed system documentation to help in system maintenance task. TRADRAD methodology, makes it easy for the programmer to write standard user documentation, having had several interactive sessions with the users in the JAD sessions and acquired a detailed understanding of the system, this invariable increases the speed of producing a well structured documentation of the system, thus improving his productivity.

System Maintenance: This stage of system development may last as long as the entire life span of the application. It also includes user support and training.

In the International Institute of Tropical Agriculture, IITA, we have a well developed programming unit which handles all the in-house development of applications needed by the institute. TRADRAD methodology has been used to implement all the applications such as the PAYPERS application which is a payroll and human resources package developed using Oracle Forms and Reports Developer running on Oracle database, interfaced with Oracle Financial Application (OFA). ; *PROMIS* - A donor and project tracking application that monitors the various projects implemented by IITA through its various donors, using Oracle Java Developer and Java Server Pages (JSP); *VEHREC* - An online Carpool Management system developed using Active Server Pages on Microsoft SQL Server. POTRACKER tracks the institutes purchasing information developed using Active Server Page as an add-on to Oracle Financial Application .

Any programming approach can be adopted, it could be the object-orientation paradigm, where each item is seen as an object with properties and classes that it inherit from a parent object or the procedural paradigm.

Various events are associated with the objects such as on mouse click, on button pressed, on mouse rollover e.t.c.

5.0 Conclusion and Recommendation

In this paper, we have looked at the features of the Hybrid system development model TRADRAD and seen how its use resulted in improving programmer productivity using the instance of System development in the International Institute of Tropical Agriculture, IITA as a case study. We also examined the issue of evaluating productivity and the concern of organisations to improve it which eventually result in the development of quality software at minimum cost to the organizations.

However, realizing that the choice of design methodology plays a major role in productive software development, it is recommended that the programmers and the stakeholders in any project development, carefully choose the best methodology to adopt in system implementation.

TRADRAD is a good hybrid methodology to adopt.

REFERENCES

Allan Rouhana: Improving speed and productivity of Software Development (2004) http://www.Programmer Productivity Research Center.htm

Clearsmith (2004): Evaluating Productivity. http://www.computer.org/productivityevaluation/icoin.htm

Donald Knuth : Literate Programming: Productivity Issues Sept. 2004. http://wwwliterateprogramming.com/

Livermore (2003): "Measuring programmer's productivity" 2003. http://www.programmer productivity.htm

Deploying Usability Engineering Methodologies (uem) for Effective Software Engineering Design Metrics

Osuagwu O.E., Chukwudebe G., Chika I.

ABSTRACT

Usability is defined as the ease of use and acceptability of a system for particular type of users carrying out specific task in a particular environment. ISO 9126 defines usability as the capacity of the software product to be understood, learned, used, and attractive to the user, when used under specific conditions. ISO/DIS 92431-11 (1996) sees usability as the extent to which a software product can be used by specified users to achieve specific goals with effectiveness, efficiency and satisfaction in a specific context of use while IEEE Std 10 (1991) refers to usability as the ease with which a user can learn to operate, prepare, inputs for and interpret outputs of a system or component. The essence of UE and User-centered Design (UCD) is to develop tools needed to support software developers in acquiring Software Engineering best practices and to develop methods for specifying end-user needs and for integrating human factors in the development process. A software that is simple to use affects positively user performance and their satisfaction, while acceptability determines the software product usability or otherwise. This paper deals with human-computer interaction and will discuss exhaustively available Usability Engineering Methods such as Heuristic Evaluation, Cognitive Walkthrough and Action Analysis under inspection method. The other usability evaluation techniques such as Thinking Aloud, field observation and questionnaire methods will also be considered. The objective of the whole exercise hinges on the belief that a good awareness of Usability Engineering Methodology will drive effectively an Information Society of the 21st century. Other UEM techniques to be discussed include Usability Context Analysis which ensures earlier critical design flaws are detected and corrected. The User Interface components such as Metaphors, Mental Models, Navigation, Interaction, Appearance and Usability as propounded by Marcus (2002) will be presented. Also to be explored include the five essential usability characteristics in software engineering projects - Learnability, Efficiency, Memorability, Low Error Rate, Satisfaction, Constructive Interaction or Co-discovery learning. Usability Engineering is User-Centred Design (UCD) technique whose objective is to specify End-User needs and integrate human factors in the Software Engineering development process. The ultimate objective of this paper therefore is to share ideas about potential and innovative ways to cross-polinate the UE and UCD methodologies, build tighter fit between Human Computer Interace (HCI), Usability and Software Engineering best practices and Research.

1.0 INTRODUCTION

The choice of usability engineering method may affect the quality and workability of particular software. It is therefore imperative that every software engineer should be aware of various usability methods and be able to quickly determine which method is most suitable to a particular software project. Usability, from our experience in human-computer interaction (HCI), must be considered first before prototyping takes place. This is because the earlier critical design flaws are detected, the more likely they can be corrected at less cost. If the flaws are detected at the end of the design cycle, changes to the interface can indeed be very costly and sometimes difficult to implement.

2.0 USABILITY CHARACTERICS

It is an accepted axiom that every good software engineering project must have five critical usability features:

- Learnability – the software should be simple to use such that the user can rapidly begin working with the system

- Efficiency: The user should be able to attain high level of productivity by the continuous use of the software.
- Memorability – The causual user should be able to return to the system after a period of non-use without having to relearn everything.
- Low Error Rate: Only easy and rectifiable error should occur when applying the software
- Satisfaction: The system should be pleasant to use.

Shneiderman (1977) has posited these criteria have tradeoff as some are more important than others; as for instance long-term efficiency may be sufficiently more important for developers to be willing to sacrifice rapid learnability than low error rate

3.0 USABILITY INSPECTION METHODS

Usability Engineering methods are divided into two broad groups: Inspection Methods whereby users are not involved and the Test methods where end users are involved.

The Inspection method identifies usability problems and improves its interface design by checking it against established standards. Its properties include heuristic evaluation, cognitive walkthroughs and action analysis.

3.1 Heuristic evaluation (HE

is the most common informal approach. It involves having usability specialists judge whether each dialogue or other interactive element follows established usability principles((Nielsen 1994). The conventional approach is for each individual evaluator to inspect the interface alone. The evaluators are allowed to communicate and aggregate their findings only after all the evaluations have been concluded. This restriction ensures independent and unbiased evaluations. During a single evaluation session, the evaluator goes through the intercace several times, inspects the various interactive elements and compares them with a list of recognized usability principles such as Neilsen's Usability Heuristics (Neilsen 1994). There are varied versions of HE currently available that have cooperative character. The heuristics must be carefully selected so they reflect the specific system being inspected, especially for Web-based services where additional heuristics become increasingly important.

The merits of this evaluation method include the application of recognized and accepted principles; intuitiveness; usability early in the development process; effective identification of major and minor problems; rapidity; and usability throughout the development process. The demerits include the separation from end users; inability to identify or allow for unknown users; inability to identify or allow for unknown users' needs; and unreliable domain-specific problem identification. Heuristic Evaluation method does not evaluate the complete design as there is no mechanism to ensure the entire design is explored, and evaluators can focus too much on one section. Moreover (Sears 1977) has queried the validity of Nielsen's guidelines.

3.2 Cognitive Walkthrough Method (CW)

CW approach is task-oriented by which the analyst explores the system'sfunctionalities, in which case CW simulates sequentially the user behaviour for a given task. CW emphasizes cognitive issues, such as learnability,by analyzing the mental processes required of the users. This can be achieved during the design by making the repertory of available actions salient, providing an obvious way to undo actions, and offering limited alternatives (Lewis 1997). CW methodology is derived from exploratory learning principles. Varied versions of CW abound including, of course, pluralistic walkthroughs wherein end users, software developers, and usability experts go through the system, discussing every single dialogue element in a software-engineering project.

The merits of CW are independence from end users and a fullyfunctioning prototype, assisting designers to take a potential user's perspective, effective identification of problems arising from interaction with the system, and the ability to help to define users' goals and assumptions. The demerits include tediousness and the danger of an inherent bias due to improper task selection. The method emphasis on low-level details and non-involvement of the end users as the project progresses.

3.3 ACTION ANALYSIS METHOD (AA)

The AA method has two broad approaches – formal (F) and back-of-the-envelop (BoE). Both emphasizes more on what the practitioners do than on what they say they do. The formal approach requires close inspection of the action

sequences a user performs to complete a task. This is what is known as keystroke-level analysis (Card 1983). It involves breaking the task into individual actions such as move-mouse-to-menu ortype-on-the keyboard and calculating the times needed to perform the actions. BoE analysis is less detailed and gives less precise results, but it can be performed much faster. This necessitates similar walkthrough of the actions a user will perform with regard to physical, cognitive, and perceptual loading. AA has the advantage of precisely predicting how long a task will take, and a depp insight into usability behaviour of the users. The demerit is that AA is time-consuming and requires considerable experience to implement.

3.4 USABILITY TEST METHODS

This is a fundamental and indispensable method in usability engineering as it provides direct information about how people use our systems and their exact problems with a specific interface. Thinking aloud (TA), Field Observation (FO) and questionnaire (Q) are the three major approaches for testing usability.

Neilsen (1994) has posited that TA may as well be the most valuable usability engineering method as it involves having an end user continuously thinking out loud while using the system. Thus, by verbalizing their thoughts, the test users enable scientists to understand how they view the system. This makes it simpler to identify the end users' major misconceptions. By showing how users interpret each individual interface item TA enhances a direct understanding of which parts of the dialogue cause the most problems. Time is critical, as the contents of the users' working memory contents are desired. The other variant of TA "*constructive interaction*" involves having two test users use a system together (co-discovery learning). The main benefits of this method is that the test situation is natural than the standard TA with single users working alone, as people are used to verbalizing their thoughts when trying to solve a problem as a group. Thus, users may make more comments when engaged in constructive interaction than when simply thinking aloud for the benefit of an experimenter. The other benefits of TA include ability to reveal why users do something, TA provides a close approximation to how individuals use the system in practice, provision of a wealth of data, which can be collected from a small number of users; user comments often contain vivid and explicit quotes. Preference and performance information can be collected simultaneously. TA assists users to focus and concentrate. Early clues helps to anticipate and trace the source of problems to avoid later misconceptions and confusion in the early stage of the software engineering design. The demerits of TA include failure to lend itself well to most types of performance measurement. Different learning style is often perceived as unnatural, distracting and strenuous by the users; non-analytical learners generallyfeel inhibited. TA is time consuming, as briefing the end users is a necessary component of the preparation.

3.5 FIELD OBSERVATION

This is the simplest Test Methods. The scientist has to visit one or more users in their work places and take notes as unobtrusively as possible to avoid interfering with their work as noise and disturbance can lead to false results. It is preferable that the observer be virtually invisible to ensure normal working conditions. Video coverage of work environment is recommended to make the observation process less obtrusive. Though this may not be very necessary, a complete record of a series of user tests can be used to perform formal impact analysis of usability problems (Holzinger 2004). Field observation is usually zeroed to major usability catastrophes that tend to be so glaring that they are obvious the first time they are observed and thus do not require repeated perusal of a recorded test session.

Data Logging (DL) is another method of digital observation, which involves statistics about the detailed use of a system and provides extensive timing data regard as important in HCI and usability. DL is applied to collect information about the use of a system after release and can be used also as a supplementary method of collecting more detailed data during user testing. An interface log will contain statistics about the frequency with which each user has used each feature in the program and the frequency with which various events of interest have taken place.

3.6 THE QUESTIONAIRE APPROACH

There are some feelings of users of a particular application, which may not be ease to extract without asking questions. Indeed many aspects of usability can best be studied by querying the

users, particularly those issues related to the subjective satisfaction of the users and their possible anxieties. Questions are excellent for studying how end users use the system and their preferred features and but design experience. Q approach is an indirect method as it does not study the actual user interface behaviour but only collects the opinions of the users about the interface. The Q method can be supplemented with oral interview, which can be designed to enable users respond and encourage elaboration of discussions. The merits of Q approach include those subjective user preferences, satisfaction, and possible anxieties can be easily identified. Questionnaires can tool for collecting and compiling statistics. The demerits is just that the method is indirect and the result has low validity and needs sufficient responsible to be significant.

4.0 SUMMARY AND CONCLUSIONS

Software Engineers need to combine Usability inspection with Usability test methods to achieve improved software usability evaluation results. For instance, a cognitive walkthrough can be augmented with a task-independent method such as heuristic evaluation. Indirect usability test such as questionnaires or interviews should be augmented with direct usability tests. What is expected is a combination of usability methods that will reveal the weaknesses of an application software user interface to enable the software designers improve its quality and usability. The ultimate objective is to build systems that users want to use!

REFERENCES

Card, S.K. et.al (1983) *The Psychology of Human-Computer Interaction*. Erlbaum, Hillsdale, NJ

Human-Computer Interaction 2nd ed. M. Helander Ed. Elsevier, Amsterdam, 717-732.

Holzinger, A. (2004) *Application of rapid prototying to the user interface development for a virtual medical campus*. IEEE Softw. 21, 1.

ISO 9126 (1991) *Software Product Evaluation: Quality Characteristics and Guidelines for their Use.*

Lewis C. And Wharton C. (1997), *Cognitive Walkthroughs. Handbook of*

Marcus A. (2002) *Dare we define user-interface design*? Interaction 9,5 19-24

Nielsen J. (1994) Usability Engineering, Morgan Kaufmann, San Francisco.

Sears, A.L (1997), *Heuristic Walkthroughs: Finding problems without the noise*.Int. J. Human-Computer Interact. 9,3, 213-234.

Selecting Tools and Methods for Usability Engineering of Interactive Systems Development

I. K. Oyeyinka, F. I. Oyeyinka (Mrs)

ABSTRACT

This paper presents an introduction to usability engineering methods that may be useful in interactive software development. It introduces the four broad categorization of usability engineering methods and goes ahead to discuss some usability engineering methods that may be used in interactive software engineering. While some of these methods are formative others are summative in nature and we conclude by recommending a combination of methods in any software engineering task.

INTRODUCTION

Usability defines the relationship between tools and their users. An effective tool will allow user to complete task in the best and easiest way possible. This principle applies to computers and software.

Today, users interface has become a very important factor in the success or otherwise of a software development project. Usability has therefore become an important aspect of system life cycle. Many practical techniques for measuring usability have been proposed to be introduced in interactive software development life cycle (Nielsen 1993). Usability engineering is a discipline that represents a melting point between software engineering and human computer interaction.

Software usability is related with making systems easy to use, easy to learn, easy to remember, pleasing, error tolerance and supportive to users during interaction with the computer equipment (Nielsen 2003). The International Standard Organization (ISO 9241-11) defined usability as the extent to which a product can be used with effectiveness, efficiency and satisfaction in a specified context of use.

WAYS OF ASSESSING SOFTWARE USABILITY

According to ISO 9241 usability of software system could be measured in three ways (Swartz K.O. 2003):

1. By analysis of the features of the product. The product features, required in a particular context may be analyzed to see if it is present and adequate.
2. By analysis of the process of interaction: This could be done by modeling the interaction between a user carrying out a task with the system. However, interaction with the system is a dynamic process in the brain and current approaches are inadequate at giving a good estimate of usability.
3. The Effectiveness and Efficiency. This could be obtained from the use of the system in a particular context. This is a direct measure of the attribute of usability.

SOME USABILITY ENGINEERING METHODS

Iterative performance of usability engineering methods is crucial to development of any application. Application development that does not include a usability strategy often results in function-rich software that is full of usability issues that constitute a barrier between users and application functionality. The practice of most software developers has been supplementing software engineering by appending a usability assessment at the end of development. This is insufficient. If usability assessment does not occur early and frequently in the development lifecycle, unnecessary resources would have been spent before the usability issues are discovered. This conflict places developers in a crossroad of deciding whether to allow known usability issues to remain in the application or redesign large sections of the already completed application interaction. Therefore, user-centered interaction design iteratively applies usability engineering methods that have been researched and proven successful in the HCI community. There are a number of methodologies used in assessing the usability of interactive applications. Deciding which usability methodology to use is based on objectives, resources, and time constraints of the evaluation. However, the ultimate

trade-off is between doing some usability assessment and none at all. (Nielsen and Mack 1994)

Although there are at least four basic categories of usability engineering methods namely automatic, empirical, formal, and inspection, only two of these categories have gained widespread use. Table 1 lists the four most common categories of usability engineering methods and a brief description of each (Swartz 2003).

Usability Method Category	Description
Automated	User interaction is described using an interface specification technique. The interface specification is then validated using a software application designed to assess usability of the interface described by the specification technique.
Formalism-based	User interaction is represented using a set of formal interaction models and then a set of usability functions is applied to the models
Inspection	Evaluations are based on a set of user interaction design guidelines and/or heuristics and rely on judgment and experience of evaluators.
Empirical	Evaluations are completed using a representative sample of application users and a pre-determined set of representative tasks (Swartz 2003).

Table 1: Usability Method Categories

Some of the methods are describe below. A combination of methods may be used in a usability evaluation task. Some of these methods presented here are formative in nature occurring early in the software usability and software engineering cycle and others are summative in nature useful at the tail end of the life cycles.

In order to achieve good product usability, tools and methods may be used at different stages in the usability engineering life cycle. A selection of pragmatic usability tools and methods, commonly used in the industry today, are listed in the table below.

To support choosing an appropriate method the table below indicates:

- The stage in the development process when they provide best results.
- The resource level required, which relates to effort and costs.
- The strength of a particular tool or method and the kind of information it provides.

Following this process to develop a product can result in a number of significant advantages for the developer, by producing products which:

- Are easier to understand and use, thus reducing training and support costs.
- Improve the quality of life of users by reducing stress and improving satisfaction.
- Significantly improve the productivity and operational efficiency of individual users and consequently the organisation.

Implementing Usability tools and methods in development life cycle

Evaluation by Inspection

This method involves four or six experts collaborating to evaluate the system in question. The significant characteristic of this approach is that it is a collaborative technique guided by a defined list of design criteria (Usability Partners).

Heuristic Evaluation

Heuristic evaluation is an expert inspection method that identifies general usability problems that users can be expected to encounter when using a product or interface. Usually at least three usability experts evaluate the system with reference to established guidelines or principles, noting down their observations and often ranking them in order of severity. It is a quick and efficient method which can be completed in a few days.

Brainstorming

Brainstorming brings together a set of experts to inspire each other in the creative, idea generation phase of the problem solving process. Brainstorming is used to generate new ideas by freeing the mind to accept any idea that is suggested, thus allowing freedom of creativity. The result of a brainstorming session is a set of good ideas, and a general insight into the solution area.

Card sorting

Card sorting techniques are used to analyze and explore the latent structure in an unsorted collection of information items, functions, statements or ideas. Each item is written on a small index card. Participants, working on their own, sort these cards into groups or clusters, the data from each individual card sort are

then combined and are analyzed statistically.

Cognitive workload assessment

Measuring cognitive workload involves assessing how much mental effort a user expends whilst using a prototype. This can be obtained from questionnaires such as the Subjective Mental Effort Questionnaire (SMEQ) and the Task Load Index (TLX) (Usability Partners).

Cognitive walkthrough

A process of going step by step through a product or system design getting reactions from relevant staff and users. Normally one or two members of the design team will guide the walkthrough, while one or more users will comment as the walkthrough proceeds.

Context of Use Analysis

Context of Use Analysis is a structured method for eliciting detailed information about a product and how it will be used. Through a workshop attended by the product stakeholders important characteristics of the users (or groups of users), their tasks, their environment are identified. Context Analysis meetings should take place as early as possible in the design of a product. However the results of these meetings can be used throughout the lifecycle of the product, being continually updated and used for reference.

Contextual Inquiry

Contextual inquiry is one of the best methods to use when you really need to understand the users' work context. It is basically a structured field interviewing method, based on a few core principles that differentiate this method from plain, journalistic interviewing. Contextual inquiry is more a discovery process than an evaluative process; more like learning than testing. This technique is best used in the early stages of development, to gain an understand of how people feel about their jobs, how they carry out their work, how information flows through the organization etc.

Cost-Benefit Analysis of Usability

Presents a generic framework for identifying the costs and benefits associated with a user-centered design activity. The first step is to identify the benefits to be gained by improving the usability of a system include increased user productivity, decreased user errors and decreased training costs. The costs of the proposed usability effort (personnel and equipment overheads) must be taken into account and subtracted from the total projected benefit value.

Focus Group

A focus group brings together a cross-section of stakeholders in a discussion group format. Views on relevant topics are elicited by a facilitator. The meetings can be taped for later analysis. Focus groups are useful early in requirements specification but can also serve as a means of collecting feedback once a system has been in use or has been place on field trials for some time. Focus groups help to provide a multifaceted perspective on requirements and identify issues that may need to be tackled.

Functionality Matrix

This process specifies the system functions that each user will require for the different tasks that they perform. The most critical task functions are identified so that more time can be paid to them during usability testing later in the design process. It is important that input from different user groups is obtained in order to complete the matrix fully. This method is particularly useful for systems where the number of possible functions is high and where the range of tasks that the user will perform is well specified.

Interactive/Computer-based prototyping

Utilizes computer simulations to provide more realistic mock-ups under development. The prototypes often have greater fidelity to the finished system than is possible with simple paper mock-ups. End-users interact with the prototype to accomplish set tasks and any problems that arise are noted.

Interview techniques

Expert and/or novice users are asked in-depth questions by an interviewer in order to gain specific knowledge, or to obtain the subjective opinions based on product usage experience. Interviews may follow a pre-specified list of items (structured) and/or may allow users to provide their views freely (unstructured). Type, detail and validity of data gathered vary with the type of interview and the experience of the interviewer.

ISO 9241 conformity assessment

Assesses a product for conformance to the relevant requirements as detailed in ISO 9241 standard: Ergonomic Requirements for work with Visual Display Terminals (VDTs). Developers provide documentary evidence regarding their development process and one or more auditors examine these documents and visit the site to interview relevant staff. Auditors determine if conformance is warranted or if not feedback is provided on non-conformances.

Paper prototyping

This method uses simple materials to create a paper-based simulation of an interface with the aim of exploring user requirements. When the paper prototype has been prepared, a member of the design team sits infront of the user and 'plays the computer' by moving interface elements around in response to the user's actions. The user makes selections and activates interface elements by using their finger for input actions. Users are given task instructions and encouraged to express their thoughts and impressions. The evaluator makes notes during the test. The method is most suitable where it is easy to simulate system behaviour or when the evaluation of detailed screen elements is not required.

Parallel Design

Several different groups of interface designers create conceptual designs within a parallel process. The aim is to develop and evaluate different interface designs before settling on a single design. The groups of designers must work independently, since the goal is to generate as much diversity as possible. Designers may not discuss their designs with each other until after they have presented their design concepts. The final result may be one of the designs or a combination of designs with the best features from each. Although parallel design may at first seem expensive, as many ideas are generated without implementing them, it is in fact a very cheap way of exploring the range of possible design concepts and selecting the probable optimum design.

Rapid Prototyping

This method develops quickly different concepts through software or hardware prototypes, and evaluates them. The rapid development of a simulation/prototype of a future product allows users to visualize it and provide feedback. It can be used to clarify user requirements. Later during development, it can be used to specify details of the user interface.

Scenarios

Scenarios are characterizations of users and their tasks in a specified context. They offer concrete representations of a user working with a product in order to achieve a particular goal. The objective of user scenarios in the early phases of development is to improve the accessibility of end user requirements and usability goals to the design team. Later scenarios can be used during design and evaluation activities.

Storyboarding

Storyboards are sequences of images which demonstrate the relationship between individual events (e.g. screen outputs) and actions within a system. A typical storyboard will contain a number of images depicting features such as menus, dialogue boxes and windows. The storyboard can be shown to the design team as well as typical users, allowing them to visualize the composition and scope of possible interfaces and offer critical feedback.

Survey

A survey involves administering a set of written questions to a large sample population of users. Surveys can help determine information or customers, work practice and attitudes. There are two types: 'closed', where the respondent is asked to select from available responses and 'open', where the respondent is free to answer as they wish.

SUMI - Software Usability Measurement Inventory

SUMI is a questionnaire designed to collect subjective feedback from users about a software product with which they have some experience. Users are asked to complete a standardized 50-statement psychometric questionnaire. Their answers were analyzed with the aid of a computer program - SUMISCO. SUMI data provides a usability profile according to five scales: perceived efficiency, affect (likeability), control, learnability and helpfulness. It also provides a global assessment of usability. The international database with which the software is being compared contains some 300 software products. SUMI can be used to assess user satisfaction with high fidelity prototypes or with

operational systems and can be used in conjunction with other usability tools and methods.

Supportive evaluation

Supportive evaluation is a participatory form of evaluation similar to a 'principled focus group'. Users and developers meet together and the user representatives try to use the product to accomplish set tasks. The designers who observe can later explore the issues identified through a facilitated discussion. Several trials can be run to focus on different features or different versions of the product.

Task analysis

Task analysis defines what a user is required to do in terms of actions and/or cognitive processes to achieve a task. A detailed task analysis can be conducted to understand a system and the information flow within it. These information flows are important to the maintenance of the system. Failure to allocate sufficient resources to this activity increases the potential for costly problems arising in later phases of development. Task analysis makes it possible to design and allocate tasks appropriately within the new system. Once the tasks are defined, the functionality required to support the tasks can be accurately specified.

Task allocation

A successful system depends on the effective allocation of tasks between the system and the users. Task allocation decisions determine the extent to which a given task is to be automated or assigned to a human user. The decisions are based on many factors, such as relative capabilities and limitations of human versus technology in terms of reliability, speed, accuracy, strength, flexibility of response, cost and the importance of successful or timely accomplishment of tasks. The approach is most useful for systems that affect whole work processes rather than single user, single task products. Designers often identify functions that the technology is capable of performing and allocate the remaining functions to users, relying on their flexibility to make the system work. This does not make best use of users abilities and skills and can lead to unsatisfactory job design.

User-based evaluation for design feedback

This method offers a relatively quick and cheap way to conduct a user-based evaluation of a current product or prototype. The focus is on task completion and the acquisition of design feedback where users are unable to complete tasks or need assistance to complete tasks. The emphasis is a few typical users as participants and detailed recordings are not essential. Observers make notes as users interact with a system to accomplish set tasks and identify the most serious user-interface problems.

User-based evaluation for metrics

This form of user-based evaluation entails a detailed analysis of users interacting with the particular system being evaluated. It is suited for evaluating either high-fidelity prototypes or functional systems. The real world working environment and the product under development are simulated as closely as possible. Observers make notes, timings are taken and video and/or audio recordings made. The observations are subsequently analyzed in detail, and appropriate usability metrics are calculated.

Video prototyping

This method allows designers to create a video-based simulation of interface functionality using simple materials and equipment. Interface elements are created using paper, pens, acetates etc. For example, using a camcorder, the movements of a mouse pointer over menus may be simulated by stopping and starting the camcorder as interfaces elements are moved, taken away and added. Users do not directly interact with the prototype although they can view and comment on the completed video-based simulation.

WAMMI - Web site Analysis and Measurement Inventory

WAMMI is an evaluation tool for web sites. It is based on a questionnaire that visitors fill out, and which gives a measure of how easy to use they think a web site is. The questions in the WAMMI questionnaire have been carefully selected and refined to ascertain users' subjective rating of the ease of use of a web site.

Wizard of Oz

Wizard of Oz is a technique used to present advanced concepts of interactions to users. In essence an expert (the wizard), possibly located behind a screen, processes input from a user and emulates system output. The aim is to demonstrate computer capabilities that cannot be done by the computer, for technical reasons or lack of resources. It is highly applicable to

"intelligent interfaces" which feature agents, advisors and/or natural language processing (Usability Partners).

CONCLUSION

A total of 30 methods are discussed here. These methods can be categorized into any of the four categories presented in table one. The methods presented here are not exhaustive but provides a good starting point for would be usability engineers and software engineers that want to incorporate usability into their software engineering process. Usability engineering is a whole field of study on its own and training in this field is recommended to all software engineers. In the future, usability engineering will be an inherent part of the expertise requirements of any software engineering project.

REFREENCES

Nielsen J. (1993) Usability Engineering, Academic Press Limited, London.

Nielsen J. (1994) Heuristics Evaluation. In Usability Inspection Methods, John Wiley & Sons Inc.

Nielsen J. & Mack R.L. (1994) Executive Summary, In Usability Inspection Methods, John Wiley & Sons Inc.

Swartz K.O. (2003) Virtual Environment Usability Assessment Methods, Virginia State University, Blacksburg, VA, USA.

Usability Partners, Selecting tools & Methods, www.usabilitypartners.se/usability (Accessed 2005).

Teaching Software Engineering: A Result-Oriented Approach

Adewumi, A.O., Okunoye B.O., Longe

ABSTRACT

A lot of efforts had been made to develop a proper curriculum for Software Engineering for undergraduate and graduate studies. Recently the National Universities Commission showed interest in setting what should be the minimum standard (course-wise) for university programmes in Nigeria. However, different groups had been set up by different international bodies in the past to make recommendations regarding software engineering curriculum. These include the Working Group on Software Engineering and Training (WGSEET) set up by Software Engineering Body of Knowledge (SWEBOK), the IEEE-CS and the ACM Software Engineering Coordinating Committee (SWECC) and the Software Engineering Education Project (SWEEP) to mention a few. The desire of our own universities is to meet up with the expected international standards in software engineering education. This paper reports a general survey of some of the recommendation and implementation for software engineering curriculum development made by different bodies and universities in developed countries with a view of enlightening on how such objectives could be achieve in our own environment. We make a few submissions aimed at developing a result-oriented approach to software engineering teaching and education. Our recommendations provide a framework for designing courses and other learning activities that will help prepare students for more advanced study and industrial challenges in software engineering.

Keywords: Software engineering, teaching and education, curriculum, curriculum development, software engineering standards.

1.0 Introduction

Ralston et al [11] defined Software Engineering as the disciplined application of theories and techniques from computer science to define, develop, deliver, and maintain, on time and within budget, software products that meet customers' needs and expectations.

This definition is however inadequate as the Software Engineering (SE) as a discipline rest on other intellectual foundations apart from that of Computer Science concepts. These are engineering knowledge complemented by the social and economic fundamentals [8].

Software Engineering is different from mere programming or with Software Management in that SE entails deliberate, collaborative creation and evolution of software-intensive systems that satisfies a wide range of technical, business, and regulatory requirements as opposed to mere implementation of functionality in programming, and managing software in an orderly, predictable fashion in Software Management.

A lot of efforts had been made to develop a proper curriculum for Software Engineering for undergraduate and graduate studies. Recently the National Universities Commission showed interest in setting what should be the minimum standard (course-wise) for university programmes in Nigeria. The feeling in many quarters is that the NUC documents is not sufficiently developed enough to meet up with the required standard for Computer Science graduates in general and Software Engineering (SE) graduates in particular. The desire of our own universities should be to set standard for SE education that will meet up with the expected international standards. Is this possible? How realistic is this goal? Is it possible to produce software engineers who would be able to meet industrial requirements as well as develop international acceptable products (software) that can compete anywhere in the world? The simple answer actually is yes. However, we must learn from what is obtained elsewhere and re-orientate our curriculum in the tertiary institutions to achieve these goals. However, we shall be pointing out in this paper that the responsibility of achieving these objectives is not only for the Federal Government (through NUC) but for professional bodies. Standards has to be set up jointly by the government as well as

experts in order fully realize our objectives.

In this paper, we try to report a general survey of some of the recommendations and implementations for software engineering curriculum development made by different international bodies with a view of enlightening on how such objectives could be achieve in our own environment.

The paper is organized thus: Section two discusses some of the fundamental problems in SE. Software engineering curriculum development is reviewed in section three. Section four presents our suggestions for a result-oriented approach for teaching SE. And the paper is concluded in section five.

2.0 The Fundamental Problems

Nowadays both industry and academia are showing a lot of interest in the Software Engineering (SE) discipline. This creates a challenge for universities to provide students with appropriate training that will prepare them for their future professional practice. Experiences over the years however have proved that this is far from being realized. Few of the fundamental problems responsible for this are highlighted below.

2.1 The Scope

Software Engineering is a very broad discipline [6]. Mary Shaw [8] also identified three core intellectual foundations that SE is based upon. These are core computer science concepts applied through engineering knowledge and complemented by the social and economic context of the engineering efforts. Ability to impact the discipline effectively within the constraints of time and resources, such as class hours, ratio of students to professors, etc., pose a great challenge.

2.2 Modeling

One of the classical difficulties in teaching Software Engineering is to transmit to students the problems and situations that take place in real projects [6]. It is very difficult to reproduce such situations in the classroom to provide a practical learning environment.

2.3 Tools

Another basic problem in our tertiary institutions is the availability of SE tools that can be used for the realization of practical work. Although, we have a number of freely distributed tools like ArgoUML on the web, these are not enough for practical training as the students are more likely to meet with professional tools like Rational in the industries which are quite expensive to acquire.

3.0 Software Engineering Curriculum Development

The fundamental purpose of any engineering degree programme is to equip students to function as engineers. This involves both acquiring knowledge and learning how to use it [3].

One important aspect of our curricula objectives therefore is to define the skills that software engineering students should be expected to develop while studying on their degree programmes. Software industry expects software engineer to successfully cope with technical challenges as well as deal with non-technical issues arising from difficult project situations. How can our current curriculum be revised to meet these needs?

3.1 The Global Nature of SE Profession

One basic problem in curricula development in SE has been the fact that some bodies or universities set curriculum that fits perfectly within the limits of their environment without considering the global nature of the SE profession. Thompson J.B and Edwards H.M. [1] submitted that software engineering is a discipline that must operate at a global level. Unlike other engineering disciplines such as mechanical engineering which developed and operates within domains defined by national and continental boundaries, SE must be viewed in a wider context. For example, software should be such that can be specified in USA, developed in India and then used globally on the Internet.

This is the goal of the International Federation of Information Processing (IFIP) of which Nigerian Computer Society (NCS) is a member. The IFIP has a mission of being a truly international, apolitical organization which encourages and assists in the development, exploitation and application of Information Technology for the benefit of ALL people. In 1994, IFIP's Technical Assembly set up a task force to consider the harmonization and acceptance of international standards for IT professional. Her executive board moved this activity on harmonization to the Technical Committee on Education which eventually produced a draft document "Harmonization of Professional Standards". The standards addressed issues such as Ethics of professional practice; Established body of knowledge; Education and Training; Professional experience; Best practice and proven methodologies; and Maintenance of competence. This

work has been acknowledged as crucial to achieving a world-wide software engineering profession [1].

3.2 The Need for Involving Professionals

We have witnessed significant advancements in the state of computer science education (and all its allied fields) for more than two decades now. A number of professional bodies have tried to encourage the development of quality and viable curricula. These include the Association for Computing Machinery (ACM), the IEEE Computer Society (IEEE-CS), and the Computer Sciences Accreditation Board (CSAB). These bodies have set up different committees and working groups to review and design standards curricula as a guide for computer science education. These groups include the Working Group on Software Engineering Education and Training (WGSEET), Software Engineering Coordinating Committee (SWECC), Software Engineering Body of Knowledge (SWEBOK), Software Engineering Education Knowledge (SEEK), etc. Some of the aims of these bodies are to foster the evolution of software engineering as a professional computing discipline; to coordinate the various software engineering "maturing" activities; to develop curriculum recommendations for software engineering programs; and to set standard upon which individuals can access their achievement, ability and competence. We could learn a lot from contribution of these professional organizations into SE teaching and curricula development. We shall discuss this more in section 4.

3.3 SE Curriculum: A Guide

3.3.1 Introductory SE Curriculum

An informal specification for the introductory part of a B.Sc. degree in Software Engineering (SE) is presented by the IEEE-CS group [5]. The specification was presented as an introductory course that provides the necessary "knowledge" and foundation a software engineering student must have. A thorough grasp of this foundational level prepares the student for higher level of competence and applications. One of the basis recommendations is that students entering the introductory SE curriculum must be competent in pre-calculus mathematics. We also present below an extract summarizing the recommendations of the WGSEET set up by Software Engineering Body of Knowledge (SWEBOK) [5]:

1. The introductory curriculum should emphasize the teaching of problem solving and critical thinking
2. Abstraction and modeling (a black box approach) should be used to teach software engineering principles.
3. Mathematics should be introduced as a software-modeling tool for small programs.
4. Software engineering principles should be introduced early in the curriculum.
5. Beginning students should be exposed to good software and programming practices, to illustrate the goals of professional software engineering. Likewise, examples of poorly developed software should be used to illustrate problems in software development.
6. Basic programming should be included as part of the introduction to software engineering. That is, software engineering principles and practices should be integrated into the teaching of programming in a high-level language.
7. Beginning students learn about and use a defined process for phased development of small programs.
8. Lectures/class discussions should be used should to teach concepts and principles, while and laboratories/tutorials should be used to teach software tools, processes, and practices.
9. Where possible the pedagogy should be "problem" driven, with multiple solutions per problem. That is, there is no "right" solution – rather, critical thinking should be applied to selecting the "most appropriate" solution.

Upon completion of the introductory curriculum with the above stated guide, students should be able to:

1. Describe the problems in software system development and evolution
2. Discuss basic concepts and practices of software engineering, in the areas of requirements, design, construction, quality, and management.
3. Model, develop, and document a modest-sized, high-quality program in a high-level programming language.
4. Apply core computer science concepts, including basic data structures, to design and implement a modest software system.

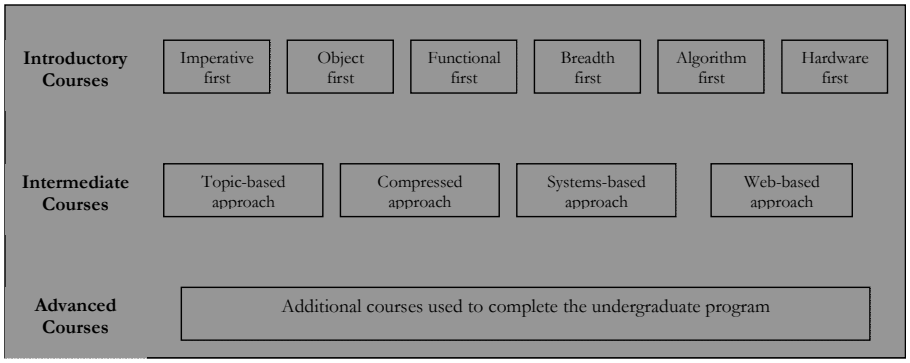


Figure 1: CC2001 Course Levels and Implementation Strategies

5. Model a software artifact's properties, and argue its correctness using sound, but informal, reasoning.

6. Appreciate basic cooperative team skills; and communicate contributions and other issues with other members, in a clear and timely manner.

7. Describe the importance of ethical behaviour and professional practice by a software engineer.

3.3.2 IEEE-CS/ACM CC2001 Curriculum Implementation Strategies

A distinction between introductory, intermediate, and advanced courses was established in the final report of CC2001 [2]. The purpose of the distinction is to provide natural boundaries for selecting implementation strategies. This report [2], for example, defines six distinct instantiations of the introductory curriculum and four thematic approaches to the intermediate courses. These implementations and their relationship in the structure of the curriculum as a whole are illustrated in Figure 1. The idea behind this structure is to offer greater flexibility by making it possible to start with any of the introductory approaches and then follow up that introduction with any of the intermediate approaches.

4.0 Towards a Result-Oriented Approach to SE Teaching

4.1 Teaching Software Development Methodology within a SE Curriculum

Software Engineering is a disciplined application of theories and techniques from computer science to define, develop, deliver, and maintain, on time and within budget, software products that meet customers' needs and expectations. The documented collection of policies, processes and procedures used by development team or organization to practice software engineering is called its software development methodology (SDM). One of the main practical aspect of SE education therefore is the teaching of SDM.

There have been recent calls by the Federal Government and other interest groups to software industry in Nigeria to develop and raise the potentials of our indigenous software experts just as it is being experienced in the developed world. This definitely will mean enhancing the software development skills of the practicing software engineers while integrating the same into the curriculum of computer science and software engineering programs. The primary aim and objective is to enhance software process productivity and quality. However, achieving this aim will mean a re-appraisal of the teaching of Software Development Methodologies (SDMs) that is currently being taught in our tertiary institutions.

Hazzan, O. and Dubinsky, Y. [10] proposed ten principles for teaching SDM in an undergraduate curriculum. Some of the principles will be useful especially in introducing new SDMs such as agile methods to students. Principle I addresses the *course* structure; Principles II-VI address the actual *teaching* of the SDMs; Principles VII-IX address the *people* involved in the educational environment – the students and the lectures/tutorial/laboratory team; the last principle – Principle X – is a *meta* principle which suggests action over preaching. (See Table 1).

The first principle suggests that any SDM should be taught as a project-based course. In such a course, students develop, within a single-semester time frame, software projects that are suited for development using the SDM in question as opposed to the generic way of presenting lectures with students passively listening to its philosophy, process, guidelines and practices.

This principle was applied in the Computer Science department of the University of Technion, United States of America while trying to introduce eXtreme Programming (XP) as a new SDM in a course titled

Table 1: Principle for Teaching SDM at a Glance

No	Category	Summary	Short Description
I	**Course**	Project-based	Course structure: Project-based team-based course
II	**Teaching**	Cognition	Teaching an SDM: What aspect to emphasize?
III		Adjustment	Adjustment of the SDM to a university course framework
IV		Projection	Projection of the SDM's notions into the project development environment
V		Connectivity	The SDM in the context of the world of software development
VI	**People**	Evaluation	Reflection of the SDM orientation in students' evaluation policy
VII		Listening	Listening to students' objections
VIII		Reflection	Students' reflections and progress diaries
IX		Feedback	Supervisors' hesitations and feedback
X	**Meta**	Inspiration	Inspiring the SDM rather than preaching it

"Operating System Projects" [10]. As reported in [10], the principles made the introduction of XP as a SDM interesting to the students who were able to successful apply it in real-life projects.

4.2 The Need for In-Course Assessment

According to Klappholz, D. et. al. [4], failure to follow best practice, rather than technological incompetence, is the cause of most software project failures. Software development requires expertise in both State of the Art (software technology) and State of the Practice (software development process). The responsibility lies therefore on computer sciences departments to assess the quality of the software development process component of their curricula and the industry to assess the efficacy of their SPI (Software Process Improvement) efforts.

Problem solving is the highest form of learning and SE is about problem solving [4]. However, knowledge alone is not sufficient for successful problem solving in a domain, the students must also choose to use that knowledge, and to monitor the progress they are making [4]. Klappholz, D. et. Al. (2003) developed an instrument for measuring Attitude Toward Software Engineering (ATSE). These have been administered to both students and software development professionals. ATSE can be used for assessing the outcome of an important aspect of the software development process component of an entire Computer Science degree program [4]. It can also be used in assessing the outcome of a course in Software Engineering, or to compare the relative efficacies of different methods of teaching Software Engineering. The ATSE survey instrument has been adapted by some faculty member teaching SE as part of their assessment and evaluation of the progress impact of their teaching on the students. Feedbacks received from such instrument administered regularly always go a long way to assess our teaching style and re-adjust where necessary to meet up to the required standard before the end of an academic session.

4.3 Creating a good Learning and Teaching Environment

Learning style has been reported to have effect on student's ability to come to grips with the concepts of programming and design [7]. As stated in the previous section, students must be giving opportunities to improve their problem solving skills and such must provide them with as much autonomy as possible. Ratcliffe, M. et.al. [7] proposed and created an integrated environment called SETLAS for teaching first year programming and design which has proved to improve student's performance and motivation [7]. SETLAS is based on the four models defined for any intelligent tutoring system namely, a knowledge model, a model of the communication media, a cognitive model of the learner and a model of the tutoring discourse (See Figure 2 adapted from [7]).

SETLAS is an automated learning environment that began with a simple objective testing system having a question bank. The aim is to provide feedback for students to monitor their individual progress. The system has the facility to e-mail students their marks after all sessions had been held. Learning objectives are set and emailed alongside with the questions to students, which is able to help them to have immediate feedback of the areas they need to revise. Students become aware also of the learning goals of the course. The statistics on performance also help the lecturer to

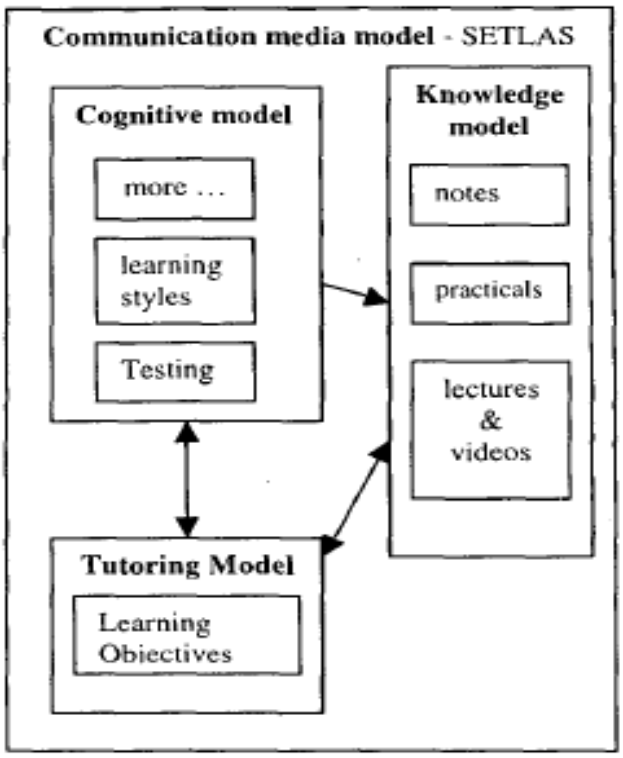
keep track of students' progress and alter his lectures where need be.

This kind of practice and tool could help raise the level of competence of SE students in the developing world. However, we must quickly state that we are limited as the enabling environment and infrastructures are not readily in place. The government as well as university authorities could therefore assist by playing more attention to proving an IT-based enabling environment for teaching practical subjects like software engineering in our universities. On the long run, the government will have lot to gain when our universities can provide enough, well-trained, capable software engineers who will raise the level our software industry.

4.4 The Need for proper Software Engineering Accreditation Exercise

The duty of accreditation of programme in Nigerian universities had lied majorly on the National University Commission (NUC). Although some profession bodies like CPN, ICAN, COREN, etc. try to accredit professional programme (under their jurisdiction) run by universities, one cannot say must have been achieved by the Computing Profession in Nigeria as regard setting minimum standards that could help in proper training of future professionals. For example, much emphasizes of the previous few accreditations were not laid on the minimum academic standards for a typical SE aspect of Computer Science. Moreover, there has not been enough publicity and acceptance of such accreditation exercise in most of our tertiary institution. The composition of the accreditation panels especially for university computing programme was questioned by many. The underscore the need for an established benchmark that set the minimum standard required for a typical SE education.

Figure 2: SETLAS Model



Another aspect of the professional body which need to be taking seriously if we are to have qualified software engineering professionals is the role of a mediator between the industry and the academia. It will assist a lot if a minimum collaboration standard is set for a typical SE programme in the universities (apart from SIWES). CPN, for example, can also assist in linking many of their registered corporate members (and others outside) with individual universities within their locality to foster technical assistance and practical development.

We provide below a number of accreditation bodies and criteria that were been set by different professional in the developed countries in order to draw some lessons from them.

The SWECC [10] released the accreditation criteria for software engineering (revised 25/09/98) and the guide to the software engineering body of knowledge, Stone Man version 0.7 [12]. The information Processing Society of Japan (IPSJ) formed in 1999 also started developing guidelines for an average curriculum model for software engineering [12]. Other accreditation criteria set by other bodies include [12]:

5 Japan Accreditation Board for Engineering Education (JABEE) Accreditation Criteria

6 Guidelines for Software Engineering Education Version 1.0

7 The CMU Master of Software Engineering Core Curriculum

8 Accreditation Board of Engineering and Technology, US (ABET) Engineering Criteria Version 2.0

9 Computing Curriculum 2001 set by ACM/IEEE-CS Joint Curriculum Task Force

10 CPN/NCS Accreditation Criteria for Software Engineering Curriculum version?

A "skill standards for information technology" was developed by the Northwest Center for Emerging Technologies (NWCET) as part of the US Department of Labour's effort to develop skill standards [12]. Professional examinations are conducted twice a year for each of the identified categories of skill labourers with the recommendation that every undergraduate

student must pass at least the examination for the first categories – the Fundamental Information Technology Engineer Examination. Thus software engineering curricula of universities are adjusted to meet the examination requirements of this category.

In the UK, the engineering profession is regulated by a set of bodies that form a hierarchical structure with two levels [3]. At the top level is the Engineering Council, which sets the generic standards for what registration (or certification). The lower level of the hierarchy consists of the various professional institutions for different branches of engineering, such as the British Computer Society and the Institute of Electrical Engineers. These administer the processes of registration and accreditation, on the basis of guidelines that interpret the requirements of the generic document prepared by the council for their particular discipline. Looking at this structure, we can then have CPN comprising of sub-bodies such as Software Engineering Sub-Committee, IT Sub-Committee made up of experts that will enforce the generic accreditation rules set up by the council to their various sub-fields.

5.0 Conclusion and Recommendations

We have reviewed the current status of our SE curriculum and the state of things with the software industry in Nigeria. We also examined the various factors that have been contributory to very successful and practical-oriented SE curriculum development in developed country which has helped their graduating Software Engineers fit easily into the industry. These factors include a practical-oriented, student friendly approach of teaching new software development methodology, a IT-based teaching environment, commitment and deep involved of well-articulated professional bodies in setting curriculum standards and/or evaluation for software engineers, and so on. We also presented an overview of the curriculum partner given by the IEEE-CS body which could serve as a guide for many of our tertiary institutions.

Our desire is to see more commitment on the part of the professional bodies in Nigeria like CPN/NCS in setting up committee/sub-committee of well experienced and learned intellectuals that will set and advocate minimum standard for both our software engineering students and practitioners with regular and more directed accreditation and examination exercises carried out to ensure compliance. Our professional bodies could also get an enabling law with the help of appropriate federal government ministries and/or legislative arms of the government to enforce compliance with this. It is our believe that we have the best of programmers and software engineers in this part of the world, we only need a little more effort to articulate things together and soonest we shall be competing with nations like India with our software industry accounting for revenues like the present crude oil.

References

Barrie Thompson J. and Helen M. Edwards. *Workshop on Achieving a World-Wide Software Engineering Profession.*

Computing Curricula (December 2001). *Computer Curricula – Computer Science Volume.* (Final Report). The Joint Task Force on Computing Curricula, IEEE Computer Society and Association for Computing Machinery.

Cowling, A. J. (2003) *What Should Graduating Software Engineers Be Able To Do?* Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03), IEEE.

David Klappholz, Lawrence Bernstein, Daniel Port (2003). *Assessing Attitude Towards Knowledge of, and Ability to Apply, Software Development Process*. Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03), IEEE

IEEE-CS (2002). *Specification for an Introductory Software Engineering Curriculum*. Report submitted by the Working Group on Software Engineering and Training (WGSEET), SWEBOK

Jorge Enrique Pérez-Martínez, Almudena Sierra-Alonso (2003). *A Coordinated Plan for Teaching Software Engineering in the Rey Juan Carlos University.* Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03), IEEE.

Mark Racliffe, Lynda Thomas and John Woodbury (2001). *A Learning Environment for First Year Software Engineers*. IEEE.

Mary Shaw (2005). *Software Engineering for 21st Century: A Basis for rethinking the Curriculum*. Institute for Software Research Interaction, School of Computer Science, Carnegie Mellon University, Pittsburgh.

Michael Gnatz, Leonid Kof, Franz Prilmeier, Tilman Seifert (2003). *A Practical Approach of Teaching Software Engineering*. Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03), IEEE.

Orit Hazzan, and Yael Dubinsky (2003). *Teaching a Software Development Methodology: The Case of Extreme Programming*. Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03), IEEE

Ralston, A.; Reilly, E. D., and Hemmendinger, D. (*eds.*) (2000) *Encyclopedia of Computer Science*, New York: Nature Publishing Group, Fourth Edition

Yoshihiro Matsumoto, Yoshihiro Akiyama, Osamu Dairiki and Tessuo Tamai (2001). *A Case of Software Engineering Accreditation*. IEEE.

* *To whom all correspondence should be addressed. Department of Computer Sciences, University of Lagos, Akoka, Lagos, Nigeria.* Email laremt_comp@yahoo.com OR laremtj@yahoo.com

Fundamental Guidelines for Conducting and Reporting Empirical Software Engineering Research (A Mini Tutorial)

Akinola, S. Olalekan, Osofisan, O. Adenike

ABSTRACT

Empirical software engineering research has been embraced for quite a long time among Computer Scientists and Software Engineers in the developed nations. To the best of our knowledge, little or nothing has been contributed by Africans, most especially the Nigerian Computer Scientists and Software Engineers in this research domain. The reason for this could be that we lack knowledge of how empirical software engineering research should be carried out in vivo and/or in vitro, among others. Empirical software engineering research needs research guidelines to improve the research and reporting process. The primary purpose of this tutorial is to introduce this 'fertile area of research' to the Nigerian Computer Scientists and Software Engineers. The tutorial covers guidelines for what to do and what not to do under six basic topics areas:

- *Experimental Context;*
- *Experimental Design;*
- *Conduct of the Experiment and Data Collection;*
- *Data Analysis;*
- *Presentation of Results; and*
- *Interpretation of results.*

Keywords: Empirical Research, Software Engineering.

INTRODUCTION

Empirical software engineering research has played a tremendous role both at industry and in academic environments. For instance, it has been used to effect changes in software development process models. New software tools and techniques are evolving in the field of software engineering. Also new quality metrics are evolving as a result of engineering research. Empirical Research, when wisely constructed and executed and when used to support the scientific method, plays a fundamental role in modern science [1].

Empirical studies take many forms. They are realized not only as formal experiments, but also as case studies, surveys and prototyping exercises as well, which could be observed in the field or in a laboratory or classroom. No matter what its form is, the essence of an empirical study is the attempt to learn something useful by comparing theory to reality and to improve our theories as a result [1].

It has been observed that many applied disciplines such as medicine, have problems performing empirical studies [2]. Many of the problems they face range from poor experimental design to inappropriate statistical techniques employed. Empirical software engineering research is also faced with these problems. To circumvent these problems, many research guidelines have been proposed for the world software engineering community to follow. Examples are the ones proposed by Dewayne et al., [1], Barbara et al [2] and Shaw [3]. These research guidelines are geared towards improving the research and reporting processes in the field of software engineering.

This mini tutorial takes after the research guidelines proposed by the above mentioned authors. In this paper, we are concerned with developing guidelines to assist interested researchers to avoid major pitfalls in their research activities and to report this research corrections

In the remainder of this paper, we describe the basic fundamental guidelines for conducting and reporting empirical software engineering research in section 2,

Section 3 gives an overview of an on-going empirical software engineering research effort in the University of Ibadan. In section 4, we give the summary, recommendations and conclusions of this paper.

2. Basic Guidelines for Conducting Empirical Software Engineering Research

It is believed that guidelines should be tailored towards particular types of study domains but most are fairly general [2]. We consider guidelines for what to do and what not to do in empirical software engineering research under six basic topic areas.

2.1 Experimental Context

The three main elements of experimental context, according to Barbara et al are

- Background information about the industrial circumstances in which an empirical study takes place or to which a new software engineering technique is developed.
- Discussion of the research hypothesis and how they were derived
- Information about related research.

There are two types of empirical software engineering studies. Observational studies (i.e. *in situ* studies of Industrial Practice) and formal experiments evaluating techniques developed in industry (e.g. inspections or design patterns).

In observational studies, there is an immense variety to be found in development procedures, organizational culture and products. Therefore, empirical studies based on observing or measuring some aspects of software development in a particular company must report a great deal of contextual information if any results and their implications are to be properly understood [2].

In addition, researchers need to identify the particular factors that might affect the generality and utility of the conclusions. Some of measurable factors based on observations are:

1. The target industries in which the products developed are used (e.g. banking, travel, petroleum, etc.).
2. The nature of the software industry (in-house or independent software supplier).
3. The skills and experience of software staff (e.g. with a language, a method, a tool, an application domain).
4. The type of software products used (e.g. a design tool, a compiler, etc.).
5. The software processes being used (e.g. a company standard process, the quality assurance procedures, the configuration management process).

Such information is essential if the same or other researchers want to replicate a study.

Formal experiments are meant for evaluating techniques developed in industry such as inspections, testing or design patterns. With this we need to be sure that the versions of the techniques that we evaluate and the outcome measures we use for testing are not over simplified. Furthermore, we need to be sure that the results are applicable in their specific circumstances.

The following are therefore proposed as empirical research context guidelines by Barbara et al [2].

1. Be sure to specify as much of the industrial context as possible. In particular clearly define the entities, attributes and measures that are capturing the contextual information.
2. If a specific hypothesis is being tested, state it clearly prior to performing the study and discuss the theory from which it is derived, so that its implications are apparent.
3. If the research is exploratory, state clearly and, prior to data analysis, what questions the investigation is intended to address and how it will address them.
4. Describe research that is similar to, or has a bearing on, the current research and how current work relates to it.

2.2 Experimental Design

This describes the products, resources and processes involved in the study including:

- The population being studied;
- The rationale and techniques for sampling from that population;
- The process for allocating and administering the treatments; and
- The methods used to reduce bias and determine sample size.

In order to select an appropriate experimental design that matches the study objectives, Barbara et al [2] proposed the following design guidelines:

1. Identify the population from which the subjects and objects are drawn.

2. Define the process by which the subjects and objects were selected (e.g. random sampling). Also discuss the inclusion criteria and exclusion criteria for your subjects.

3. Define the process by which subjects and objects are assigned to treatments. Allocation of subjects should be done in an unbiased manner or else the experiments will be compromised.

4. Restrict yourself to simple study designs, or, at least, to designs that are fully analyzed in the statistical literature. If you are not using a well-documented design and analysis method, you should consult a statistician to see whether yours is the most effective design for what you want to accomplish.

5. Define the experimental unit (individual within the organization or the organizations as a whole?).

6. For formal experiments, perform a pre-experiments or pre-calculation to identify or estimate the minimum required sample size.

7. Use appropriate levels of blinding such as blind allocation of materials in which the procedure for assigning subjects to treatment groups is separate from the process by which subjects are given any of the materials they will use during the experiments; or by blind marking in which we encode the materials for the experiment such that the markers cannot tell to which subjects or to which treatment group a particular outcome script belongs; or by blind analysis in which the treatments are coded, and the analyst does not know which treatment group is which.

8. Avoid the use of controls unless you are sure the control situation can be unambiguously defined.

9. Fully define all treatment if experiments are to be replicated and/or their results are to be taken up by industry.

2.3 Conducting the experiment and Data Collection

The process of conducting an experiment involves collecting the experimental outcome measures. This is a particular problem for software experiments because our measures are not standardized. Therefore, the data collection process must be well defined for our experiment to be replicated. The following data collection guidelines must be strictly adhered to:

1. Define all software measures fully, including the entity, attribute, unit and counting rules. Measures must be defined carefully enough so that we can understand the differences in measurement and thus, can determine whether we can translate from one measurement scheme to another. For example, some practitioners measure effort in hours while others measure it in days.

2. Describe any quality control method used to ensure completeness and accuracy of data collection.

3. For surveys, monitor and report the response rate and discuss the representativeness of the responses and the impact of non-response. It is important to determine the response rate, but it is even more critical to ensure that the non-responders have not biased the results.

4. For observational studies and experiments, record data about subjects who drop out from the studies.

5. For observational studies and experiments, record data about other performance measures that may be affected by the treatment, even if they are not the main focus of the study. For instance, in medical pharmaceutical studies, it is important to record all adverse effects of drugs under test. In software engineering, many of our outcome measures (defect rates, productivity, lead time) are related to each other. Thus, if our main interest is in whether a development method decreases lead time, it may still be important to investigate whether it has adverse or beneficial effects on productivity or defect rates.

2.4 Data Analysis

There are two approaches to analyzing experimental results [2]: Classical analysis (often referred to as the "frequentist" approach) and the Bayesian analysis, which provides a systematic means of making use of "prior information". However, Bayesian methods are not usually used in software engineering studies. We need to consult statisticians if we actually want to use it.

Other forms of data analysis are parametric and non-parametric analysis. If the distribution of variables can be identified, appropriate parametric tests are preferred. Basically, the data should be analyzed in accordance with the study design. Further guidelines on this could be obtained from the work of Barbara et al. [2].

2.5 Presentation of Results

Presentation of results is as important as the analysis itself. The reader of a study must be able to understand the reason for the study, the study design, the analysis, the results and the significance of the results. Not only do readers want to learn what happened in a study, but they also want to be able to reproduce the analysis or even replicate the study in the same or a similar context. The following guidelines as presented by Barbara, et al [2] will be of help in regard to this issue:

1. Describe or cite a reference for all statistical procedures used.
2. Report the statistical package used.
3. Present quantitative results as well as significance levels. Quantitative results should show the magnitude of effects and the confidence limits.
4. Present the raw data whenever possible. Otherwise confirm that they are available for confidential review by the reviewers and independent auditors.
5. Provide appropriate descriptive statistics.
6. Make appropriate use of graphics.

2.6 Interpretation of Results

The main aim for the interpretation or conclusions section of a paper is that any conclusions should follow directly from the results. Thus, researchers should not introduce new material in the conclusions section.

1. Define the population to which inferential statistics and predictive models apply.
2. Differentiate between statistical significance and practical importance. Research may show a statistical significance in some result, but there may be no practical importance. Confidence intervals can help us in making this determination, particularly when statistical significance is small
3. Define the type of study: observational or experimental.
4. Specify any limitations of the study. Discuss the threats to internal and external validities as relate to your research work.

3. Empirical Software Engineering Research Effort at the University of Ibadan.

The Department of Computer Science, University of Ibadan, established in 1974 has been putting efforts in the field of software engineering research both at undergraduate and postgraduate levels in recent years. Software Engineering as a course is part of undergraduate and postgraduate courses in the department. Students begin to learn the software engineering techniques right from their third year (300 level) up to the postgraduate level. 300 level undergraduate students in 2003/04 recently took up a research to understudy the software industry in Nigeria (Lagos, to be specific since we believe most of the software houses are sited at Lagos). Their findings reveal that out of 11 distinct software houses surveyed, many of them are into general application software development, none into system software development, many of the projects being undertaken by them are unsuccessful due to unrealistic cost, good programmers who can turn out codes in days are their priority, etc.

At postgraduate level, a PhD research is ongoing by the student author of this paper to understudy the factors affecting effectiveness of software inspections at industry level. This same topic had been understudied in the student academic environment in which was discovered that inspection team sizes, effort and inspection methods (preparation and meetings) greatly affect the effectiveness of software inspection.

A software engineering research group was formed recently in the department to foster the development of empirical research in the University and the nation in Nigeria.

The group has recently identified problems of conducting industrial software experimental research in Nigeria among which are:

(i) Funding

(ii) Software practitioners resisting being measured

(iii) Software compromise in which many of the important steps in software developments are being compromised for programming by software developers.

(iv) Problems of meeting deadlines when experiments are/designed around the software development process of industries and a host of others.

However, efforts are being made to liaise with some software houses in the area of conducting observational studies on their processes so that their development patterns could be understudied for possible improvements.

4. Conclusion and Recommendations

We have highlighted several guidelines as presented by Barbara, et al [2] that we hope will lure the Nigerian computer scientists and software engineers to the field

of empirical software engineering research in Nigeria. We also hope it will improve the quality of performing and reporting empirical research in software engineering for those who are already in the field.

We hereby conclude with these recommendations:

Nigerian computer science and / or software engineering community should endeavour to embrace empirical software engineering research at this point in time. This is a 'fertile' area of research that has been embraced by western scientists for a long time. A lot of abroad-based grant organizations are willing to sponsor empirical software engineering studies, most especially if most of the experiments that have been done can be reproduced in Africa.

It is also a high time for Nigerian software industry to embrace the research efforts of academics in their software development processes. A lot of benefits can be gotten from this. New tools and techniques can be brought up for them in the course of experiments.

5. References

1. Dewayne, E. Perry, Adam A. Porter and Lawrence G. Votta, Empirical studies of software engineering: A roadmap, in *Proceeding of the 22nd Conference on Software Engineering,* Limerick Ireland, June 2000.

2. Barbara A. Kitchenham, Shari Lawrence Pfleeger, David C. Hoaglin, Khaled El Emam and Jarrett Rosenberg, Preliminary Guidelines for empirical research in software engineering, *IEEE Transactions on Software Engineering,* vol. 28, no. 8, p. 721 - 734, August, 2002.

3. Mary Shaw, Writing good software engineering research papers, a Mini tutorial, Carnegie Mellon University.

Requirements Engineering Techniques For Building Internationally Functional Software

I.K. Oyeyinka, A.O. Itegboje

ABSTRACT

Software Engineering involves using disciplined methodology in all stages of program development. Software intended for international audiences must exhibit functional and non-functional features appropriate to end users culture and convention (Mahemoff and Johnston 1999)*. This paper discusses the issues involved in developing software for international market. This involves two levels; internationalization and localization. Software internationalization starts with separating requirements that are culture specific from the general one. Cultural factors can be categorized into overt factors and covert factors. Overt factors affect non-functional requirements while covert factors affect functional and non-functional requirements. The cultural factors must be understood in other to be able to build user bases for requirements specification for building internationally acceptable software.*

Introduction

Software engineering involves using disciplined methodology in all stages of program development. This sequence of stages from conception to operation of a program is referred to as software life cycle. There are five major phases of software life cycle; requirement analysis, design, coding, testing, operation and maintenance (WU 2001). At the requirement analysis stage it is important that all the features of a program are carefully specified and in a manner that are testable.

Moreover, software intended for international audiences must exhibit functional and non-functional features which are appropriate to the end users culture and conventions (Mahemoff and Johnston 1999). These cultural factors impact on requirements and must be specified appropriately to enable the resulting program be useful to a wide range of users.

Software Internationalization

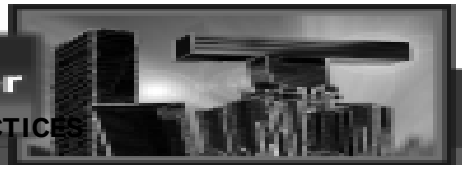
Increasing connectivity, globalization and increasing use of technology are making software developers to be concerned with the internationalization of their products (Mahemoff and Johnston 1999). There are many advantages in developing software that can be used by a wide range of people. Users benefit by getting software that closely matches their needs. As nations draw closer, there are great benefits in using the software that other use. To the organization, there is an opportunity to boost international market share while reducing risks through the widening of their market.

Implementation of software for internationalization involves two levels; Internationalization and localization. Internationalization prepares the software system to enable localization and localization is the process of plugging in the locale-specific data into the internationalized structure (Mahemoff and Johnston 1999). At the internationalization stage, there is a need to distinguish between the requirements which are common to all users and those that are culture specific. Those features that depend on the needs of end users are customized while other parts of the system will be common for all versions.

It is rather not an easy task separating requirements that are culture specific from the ones which are general. One way of doing this is through prototyping and beta testing. However, errors revealed at the testing stage are always difficult or costly to correct (Lim and Long 1994). We shall discuss how cultural factors can be categorized with the intent of helping a software developers identify culture-specific information and use them during requirements specification.

Categories of Cultural Factors that Affect Software Design

Culture extends beyond ethnicity to include factors like hobbies and lifestyle. This implies that people can belong to more than one culture. Hence culture usually goes across geographical boundaries. Locale is a term used to refer to the realization of cultural variables in software. Uren et al (1993) reported in Mahemoff and



Johnston (1999) defined locale as a combination of language, region code and character set.

Software that were built without a detailed requirements analysis of the users' culture may result in usability problems. Hence a model of culture should be used in requirements analysis. There are many models of culture that have been proposed in literature but Yeo (1995) model of culture is found to be more relevant in software requirements specification. Yeo (1995) proposed a model of culture which he categorized into overt and covert factors.

Overt Cultural Factors

Overt factors are observable features of culture like the calendars, measuring units, character sets and codes. These are daily conventions which the society operates and must be supported in software. Overt factors most often vary according to national boundaries.

According to Mahemoff and Johnston (1999), overt cultural factors can be sub divided into the following:

- Time factors like the calendar, day turnover etc.
- Writing issues like character set, text direction, special character etc.
- Language like word ordering, use of jargon etc.
- Measures like currency, unit of length, liquid measure etc.
- Formatting like number, date and time, number rounding etc.
- External system like standard paper size etc.

The users should be able to customize software according to these factors.

Covert Cultural Factors

Covert factors are those aspects of the society that are easily misunderstood by outsiders and are sometimes so subtle that they may not be noticed. Examples of covert factors include verbal and non-verbal communication style, symbols meaning and problem solving techniques (Mahemoff and Johnston 1999).

Covert factors include the following:

- **Mental Disposition**

  Mental disposition differs from culture to culture and this may impact on software acceptability and usage. For instance, Evers and Day (1997) reported that Indonesians judged software predominantly on the basis of usability while the Chinese looks more on the usefulness of the software. Another example as reported by Ito and Nakakoji (1996) asserted that Japanese regard trial and error as time consuming and tedious. Hence the issue of reversible functionality takes lower priority in Japanese software.

- **Perception**

  People's interpretation depends on their experience. Perception problems may arise from differences in interpretation rather than program flaws. For example, Russo and Boor (1993) reported in (Mahemoff and Johnston 1999) comment that red signifies danger in America and happiness in China. Hence a red icon might convey different moods to American and Chinese users of the same software.

- **Social Interactions**

  Human-human interactions vary with culture. Body language, facial expression and dialogue styles depend on everyday social interactions. Different hand postures have different meanings in different cultures. Hence an animated agent representing a hand should be designed considering the culture of the users. Social rules familiar to the human agent should be obeyed by the computer agent in human–computer interaction. Belge (1995) reported that it is not polite to beep in Japan since it will attract attention to possible error by the users.

- **Context of Use**

  The environments in which users operate vary and impose constraints on requirements. For example Americans work in enclosed areas while Japanese work in open office (Fenandes 1995). While e-mail facilitates Americans jobs, Japanese has less need for e-mails but he is open to embarrassment from beeps after errors (Mahemoff and Johnston 1999).

  Covert factors must be identified early in software project or they may cause functionality problems for developers.

Developing Culture-Enable Requirements Specification

It is necessary but not sufficient to classify users by their system-oriented roles like administrator, advanced users, data entry operators etc. There is a need to understand variables across markets. This will enable requirement analyst examine user characteristics to forecast user bases for system improvement.

There are differences in the way Overt and Covert cultural factors affect requirements. Overt factors usually impact directly on non-functional requirements, for example, the use of metric or imperial format means that the software must support their preferred measurement format. This is more of customization than internationalization. On the contrast, Covert factors may impact on functionality, for example, user interfaces such as images, color schemes, and auditory cues may be affected.

While it is necessary to understand the various cultural factors, it is more difficult to develop them into a requirements specification. However, if these factors are not properly treated, they may lead to insufficient, unusable and offensive software to people of other cultures. The use of local experts, prototyping and tests can help developers acquire and support internationalization issues adequately.

Conclusion

We have discussed how cultural factors affect requirement specification. Establishing requirements for users of other cultures requires the understanding of such culture. We have distinguished between overt and covert factors of culture. Overt factors such as measuring unit, calendar are easily understood. Covert factors such as mental disposition and perception require special consideration because they are usually hidden. Local experts, prototyping and tests are useful in constructing user bases of culture aware requirements specification.

References

Berge, M (1995). The next step in Software Internationalization. Interactions 2(1)

Evers, V. and Day, D (1997). The role of culture in Interface acceptance. Human- Computer Interaction: Interact 97, 260-267, Chapman and Hall, London.

Fernandes, T. (1995). Global Interface Design. AP Professional, Chesnut Hill, MA.

Ito, M and Nakakoji, K. (1996). Impact of culture on user Interface Design. International User Interfaces. John Wiley and Sons, NewYork

Lim, K. and Long, J. (1994). The MUSE Method for Usability Engineering, Cambridge University Press, Glasgow.

Mahemoff, M.J. and Johnston, L.J. (1998). Software Internationalization: Implications for requirement Engineering, Proceedings of the Third Australian Workshop on Requirements Engineering, 83-90. Deakin University.

Russo. P and Boor, S (1993). How fluent is your Interface? Designing for International Users. InterCHI 1993, 342-347 IOS Press, Amsterdam.

Uren, E., Howard, R., and Perinotti, T (1993). Software Internationalization and localization. Van Nostrand Reinhold, NewYork.

Wu, C.T. (2001). An Introduction to Object Oriented Programming with Java. 29 - 31, McGraw-Hill Companies, China.

Yeo, A. (1996). World-wide CHI: Cultural user Interfaces, a silver lining in cultural diversity. SIGCHI, 28(3), 4-7.

Digital Rights Management Technologies and Intellectual Privacy

Longe Olumide Babatope

ABSTRACT

Digital Rights Management (DRM) technologies have emerged as tools for the control of access and usage of digital files over computer networks and the Internet in particular. To achieve the above objectives, DRM systems are designed with the ability to monitor what is being done with the information obtained from the Internet. They can also constrain users regarding copying and modifying information to which they have access. As a result of their mode of operation, concerns are now increasing over the role of DRM systems as a possible means of intruding into the private activities of users of digital contents since they have the potential for easy collection of information about user' intellectual, social-economic and psychological preferences. This paper attempts to elucidate the privacy interests enjoyed by individuals when engaged in intellectual activities and the possible negative impact that DRM technology implementations such as monitoring, constraints and self-help features could portent for privacy rights. It advocates for a balance between technology implementation, protection of author's right's and the right to privacy in the individual domain in the design, development and implementation of DRM technologies.

Keywords: Autonomy, control, constrains, DRM systems, intranet and intellectual privacy.

1.0 Introduction

Obtaining information from the Web or an intranet and accessing an e-mail box in the privacy of one's home gives users a sense of control and psychological distance. There is also a sense of privacy and autonomy over one's learning as hypertext removes the restrictions inherent in linear text. However, both informed and uninformed users of digital data in business, academic, sports, news, entertainment and the society at large are in the center. Those with the right facilities can use the latest management tools to track every connection or simply observe without being seen.

Developments in digital trailing system have advanced to an extent that they can now be used to track the time users spend online, noting log on and log off. It enables the observer to calculate the time taken to perform a task, to note laziness, to evaluate the aptitude of the observed, to judge performance, to assess success, to monitor the nature and type of the consumed content and to classify the observed. It all boils down to the fact that the sense of autonomy, anonymity, privacy and control over one's intellectual and social consumption on the Internet is an illusion!

1.1 The Illusion of Privacy

When we use shared folders, download assignments, and browse the Internet, data are gathered directly and indirectly about us. From what feels like privacy, we participate in on-line chat rooms, leave a message on a bulletin board, shop online, and register with a commercial site. With more classes and degrees offered online, education is becoming consumer oriented. Knowledge is now considered a content commodity. Students have a sense of being in control with access to information anytime and anyplace. Ironically, users are more visible and open to surveillance (Introna, 1999). New technologies called "ubiquitous computing" now include the ability to track an employee's location at all times.

The threat from invasion and unathorised usage of digital content over the Internet is very real. Content providers can provide authorization and access control to ensure that only paying users can access content. They can also use encryption to protect content during transport. The major challenge therefore is how to control what customers do with the purchased content once it reaches their premises. As a result of the limitations involved in controlling what customers do with Internet contents of all types, content owners are reluctant to release valuable facts in digital format because they fear for the unauthorized usage of their content (What is now popularly referred to as napsterization) (John, 2003).

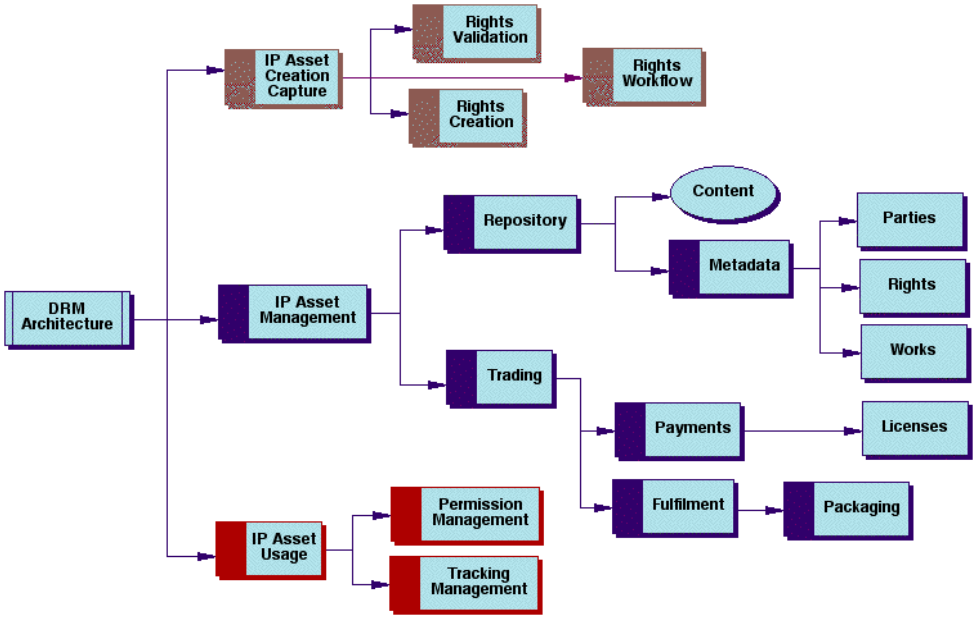


Figure 1 - DRM Functional Architecture (Source: Renato I. (2001)

2.0 Digital Rights Management (DRM) and Privacy Implications

DRM (Digital Rights Management) refers to technologies and services that enable legitimate owners of intellectual property to regulate the right of access to their assets via electronic means. There is a strong relationship between digital products and the Internet. The network provides a vibrant market for digital products; however, once stolen, digital products can be broadcast quickly to all corners of the earth. According to Moffaert (2003) the term "Digital Rights" refers to copyright and related rights of digital content (data, audio and video). Digital Rights Management is sets to achieve the following goals:

- Identify digital rights;
- Describe associated usage rules and
- Enforce these usage rules using Digital Rights Management Software.

In designing and implementing DRM systems, there are two critical architectures to consider. The first is the functional architecture, which covers the high-level modules or components of the DRM system that together provide an end-to-end management of rights. The second critical architecture is the information architecture, which covers the modeling of the entities within a DRM system as well as their relationships (Hoeifmester, 2000).

2.1 Functional Architecture

The overall DRM framework suited to building digital rights-enabled systems can be modeled in three areas:

- **Intellectual Property (IP) Asset Creation and Capture:** How to manage the creation of content so it can be easily traded. This includes asserting rights when content is first created (or reused and extended with appropriate rights to do so) by various content creators/ providers.
- **IP Asset Management:** How to manage and enable the trade of content. This includes accepting content from creators into an asset management system. The trading systems need to manage the descriptive metadata and rights metadata (e.g., parties, usages, payments, etc.).
- **IP Asset Usage**: How to manage the usage of content once it has been traded. This includes supporting constraints over traded content in specific desktop systems/software.

While the above models comprise the broad areas required for DRM, the models need to be complemented by the functional architecture that provides the framework for the modules to implement DRM functionality.

2.2 The Privacy Implications

Some philosophers conceive of "privacy" as a condition of inaccessibility or limited accessibility to the rest of the world (Anita, 1988). Invasions of privacy involve rendering the individual more accessible to others in some way. The future of privacy is increasingly linked to the future of copyright enforcement. In an effort to control the proliferation of unauthorized copies, and to maximize profit from information goods distributed over the Internet, copyright owners and their technology partners are designing digital rights management ("DRM") technologies that will allow more perfect control over access to and use of digital files.

Unfortunately, the same capabilities that enable more perfect control also implicate the privacy interests of users of information goods. Although DRM technologies vary considerably, at the most general level they represent an effort to reshape the practices and spaces of intellectual consumption. They also create the potential for vastly increased collection of information about individuals' intellectual habits and preferences. These technologies therefore affect both spatial and informational dimensions of the privacy that individuals customarily have enjoyed in their intellectual activities (Julie, 2003)

3.0 Strands of Privacy Theory

Two basic views of the privacy theory inform the concept of the individual interest in intellectual privacy. These strands converge to define a zone of privacy for intellectual activity that has physical as well as conceptual dimensions. Specifically, the individual interest in intellectual privacy extends both to information about intellectual consumption and exploration and to the physical and temporal circumstances of intellectual consumption within private spaces. As conventionally understood, interests in intellectual privacy derive from interests in personal autonomy, and are primarily informational (James, 2003).

Surveillance and compelled disclosure of information about intellectual consumption threaten rights of personal integrity and self-definition in subtle but powerful ways. Although a person cannot be prohibited from thinking as he/she chooses, persistent, fine-grained observation subtly shapes behavior, expression, and ultimately identity. Additionally, surveillance and exposure devalue the fundamental dignity of persons by reducing the exposed individuals to the sum of their "profiles." This is the case in circumstances where records of intellectual consumption such as libraries usage, visits to web sites, access to academic journals, video rental memberships, and cable subscriptions are routinely generated (Introna, 1999).

The second strand of privacy theory that relates to intellectual privacy concerns privacy within physical spaces. Tradition and social practice reserve certain types of "private space" to the individual or the family. Chief among these is the home, which is conceived as a place of retreat from the eyes of the outside world. Not every invasion of a residential property interest is an invasion of privacy. For example, most people do not think that a nuisance, such as excessive noise or noxious fumes, is also a privacy invasion. Individuals can have privacy expectations in spaces that they do not own or rent, such as public restrooms, dressing rooms, and telephone booths. Acknowledgment of these expectations suggests a fairly broad consensus that the interests protected by "privacy" and "property" is different (Julie, 2000).

Among the behaviors shielded by spatial privacy are those relating to activities of the mind. Just as spatial privacy allows for physical nudity (within one's

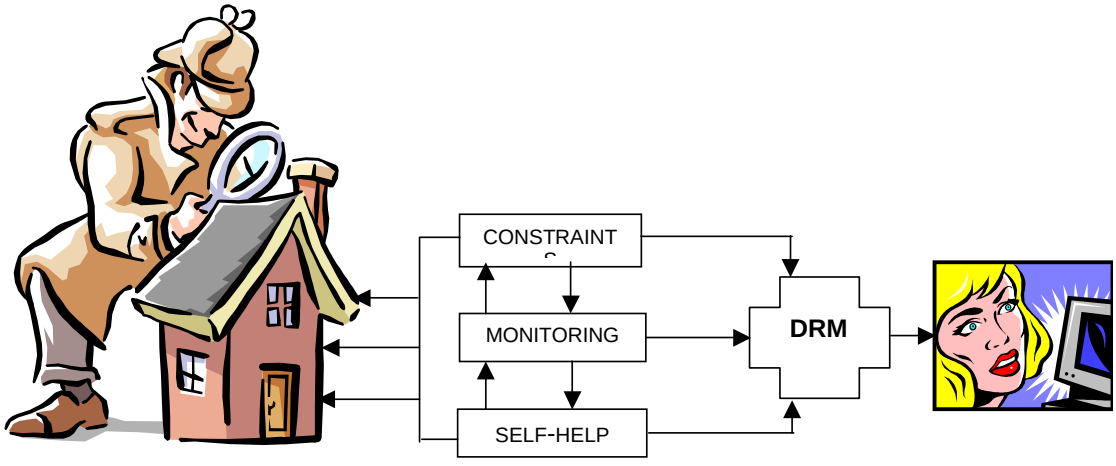


Fig 2.0: Surveillance Using DRM Technologies

bedroom), so it also allows for metaphorical nudity; behind closed doors, one may shed the situational personae that one adopts with co-workers, neighbors, fellow commuters, or social acquaintances, and become at once more transparent and more complex than any of those personae allows. Spatial privacy affords the freedom to explore areas of intellectual interest that one might not feel as free to explore in public. It also affords the freedom to dictate the circumstances i.e. when, where, how, and how often one's own intellectual consumption is, provided it is unobserved and unobstructed by others.

3.1 DRM Technologies and Intellectual Privacy

Despite all the protection it affords owners of contents, DRM technologies are poised to affect both the spatial and the informational dimensions of intellectual privacy. By directly constraining private behaviors related to intellectual consumption and by enabling creation of detailed and permanent records of such consumption, these technologies have the potential to change dramatically the way people experience intellectual goods. Summarised below are the three possible ways in which this will be done:

3.1.1 Constraint

DRM technologies are designed to set and enforce limits on user behavior. For example, a music delivery format might prevent copying, including copying for future consumption, or might restrict the types of devices that can be used for playback. The "Content Scrambling System" (CSS) algorithm used on DVDs does both of these things, and also implements a "region coding" compatibility system designed to ensure that DVDs intended for use in one geographic region cannot be played on equipment sold elsewhere (Longe, 2004)

Technologies that constrain user behavior narrow the zone of freedom traditionally enjoyed for activities in private spaces, and in particular for activities relating to intellectual consumption within those spaces. In so doing, they decrease the level of autonomy that users enjoy with respect to the terms of use and enjoyment of intellectual goods. This restrictions, to say the least, amount to an invasion of privacy.

3.1.2 Monitoring

Some DRM technologies are designed to report back to the information provider on the activities of individual users. Such reporting may occur in conjunction with a pay-per-use arrangement for access to the work, or it may occur independently of payment terms. For example, monitoring functionality might be designed to collect data about use of the work that might reveal user preferences for particular types of content.

Monitoring can also be used to determine information about related products, such as the presence of non-copy-protected MP3 files on the user's hard drive or other computer programs a user is running in conjunction with a licensed program. DRM technologies that monitor user behavior create records of intellectual consumption. Indirectly, they also create records of intellectual exploration, one of the most personal and private of activities thereby producing records of behavior within private spheres. Spheres within which one might reasonably expect that one's behavior are not subject to observation.

Gathering information about intellectual consumption renders intellectual preferences accessible, both to the information provider and to third parties that might purchase it or invoke legal process to compel its production. To the extent that behaviors within private spaces become accessible, or potentially accessible to the outside world, the individual has lost a portion of the privacy that seclusion ought to guarantee. Absence of stringent privacy protections and the threat of disclosure may discourage intellectual exploration, and therefore compromise intellectual privacy interests (Jeff, 1999).

3.1.3 Self-Help

Direct restriction protocols can be designed to encode penalties as well as disabilities. For example, a DRM system could be designed to disable access to a work upon detecting an attempt at unauthorized use. Such "self-help" technologies (designed to obviate recourse to legal enforcement procedures) might be directed and controlled externally upon detection of prohibited activities. This type of functionality would need to be implemented in conjunction with monitoring functionality (Chris, 2002). Self-help technologies might operate automatically upon internal detection of a triggering activity, without communication with any external system or controller.

DRM self-help technologies present a special case of the constraint problem, and potentially a special case of the monitoring problem as well. The punitive quality of self-help implicates privacy interests in one way that technologies of direct constraint do not. The identification of a particular consumer as a target for self-help measures entails loss of the relative anonymity formerly enjoyed by that individual as one among many customers. (James, 2003).

3.2 The Legal System and the DRM Challenge

Many aspects of law bear to some degree on questions of intellectual privacy, but none is exactly developed to address the unique privacy problems created by DRM technologies. Several, however, have the potential to do so. The common law of privacy, with its emphasis on control over personal spaces, private facts, and commercialization of image, can be reconfigured for the digital age by drawing on the policy and normative frameworks embodied in other privacy-regarding areas of law. In addition, because many information goods are also consumer goods, a more explicitly regulatory approach to privacy-invasive DRM technologies, grounded in principles of consumer protection law, can significantly improve levels of protection for intellectual privacy (Samuel & Louis, 1990)

The argument that effective privacy protection should include control over the spaces of intellectual consumption finds support, instead, within both the substantive provisions and the overall structure of copyright law. The fair use doctrine, which sanctions certain acts of private copying, shields a range of actions that users might take in private spaces, including time and space-shifting of copies, loading and reloading of digital files, and manipulation of digital content. Also the first sale doctrine, which establishes the right to dispose of one's copy of a work without any obligation to seek the copyright owner's approval similarly rests, on the belief that a copyright owner has no cognizable interest in a broad range of post-purchase user activities or in the spaces where they occur (NIST, 2003).

More broadly, because copyright law does not give copyright owners the exclusive right to control all uses of their copyrighted works, it implicitly reserves to users the right to engage in conduct not encompassed by copyright laws. For example, copyright encompass such acts as reading a copy of a book, viewing a copy of a movie, or listening to a copy of a musical recording that one owns; not coincidentally, these are all acts that ordinarily occur within private spaces (Longe, 2004).

4.0 Balancing Social and Technological Impact in the Design of DRM Systems.

The notion of balanced design is an outgrowth of the interdisciplinary study of science, technology, and society. Careful attention to the social embeddedness of technologies reminds us that technologies themselves are social artifacts; they constitute and are constituted by social values and interests (Batya et al, 2002). This insight, in turn, suggests that careful attention to values and value choices at the design stage might produce important payoffs. A technical investigation of the range of design possibilities, and empirical study of user experiences with responses to different designs options would do DRM architects a lot of good.

A balanced design process for DRM technologies would seek, among other things, to create rights management infrastructures for information goods that respect and seek to preserve user privacy (Dan & Julie, 2001). Such infrastructures would have three components, which maps to the three types of DRM functionality discussed earlier. Technically, then, the challenge lies in developing systems that preserve both enough privacy for users and enough control for rights owners. One example of such a technology is the serial copy management system mandated by the Audio Home Recording Act, which allows the production of perfect first-generation copies but causes significant quality degradation in subsequent generations. Another example is the Digital Millennium Copyright Act's (DMCA's) requirements that analog videocassette recorders be designed to allow consumers to time-shift some television programming (Wurzler, 2004). A careful, iterative methodology, incorporating participation by the full range of interested parties could help designers negotiate the challenges entailed in implementing planned imperfection.

Value-sensitive design for DRM also would investigate methods of building in limits on monitoring and profiling of individual users. Since most businesses need to collect and retain some information about their customers to manage orders, payments,. and deliveries, technological limits on data collection and use cannot fully substitute for other, human-implemented safeguards. Nonetheless, DRM systems may be designed either to minimize or to maximize data collection, retention, extraction and use. To preserve the intellectual privacy of information users, DRM design should incorporate minimization principles.

Finally, a value-sensitive design approach to DRM technologies would consider the desirability of implementing limitations on self-help. For example, after weighing the full spectrum of values implicated by automated, enforcement actions, designers might conclude that digital content files should never be programmed to self-destruct, or to deny access entirely, upon detecting impermissible actions by users. Alternatively, they might conclude that denial of access should be permissible, but only in certain clearly defined and extreme circumstances.

5.0 CONCLUSION

For all of the reasons already discussed, it is analytically sound to conclude that DRM technologies have the potential significantly to diminish privacy in intellectual

consumption. It is worth noting, that the deployment of DRM technologies of self-help, and more generally of constraint, also raises questions about the nature and function of the boundary between public and private spheres. DRM technologies may represent the future of information access and use, but their design and implementation are still open questions. A shift to an information environment characterized by pervasive constraints, universal monitoring, and automated self-help would severely undermine intellectual privacy values.

Instead, in the era of DRM, law and technology together must share responsibility for protecting intellectual privacy. Law can fulfill its responsibility by defining individual rights and correlative obligations. Technology will ensure that its designers and their customers in the content industries practice both inclusiveness and restraint, but to do so effectively they must come to terms with the importance of law, and more broadly of public policy and public values, in establishing design parameters.

REFERENCES

Anita, A. (1988): Uneasy Access: Privacy for Women in a Free Society. Stan. Law Rev. Vol. 52, No. 28

Batya, F., Daniel, C. & Edward, F. (2002): Informed Consent in the Mozilla Browser: Implementing Value-Sensitive Design. Proc of the 35th Hawaii Int. Conf. on System Sciences. Available online at http://dlib2.computer.org/conferen/hicss/pdf

Hoeifmester, D. (2000): Applied Software Architectures. C Hofmeister, R Nord, & D Soni. Addison-Wesley,.

NIST(2003): Digital Rights Management: Establishing an Infrastructure for a Digital Economy

Available online at http://www.itl.nist.gov/div895/whatdoes.html

Renato, I. (2001): Digital Rights Management (DRM) Architectures. D-Lib Magazine. Vol. 7 No. 6

Chris, J. (2002): Consumer Privacy in the E-Commerce Market-place 2002, Internet & Business Journal. Available online at http://www.epIc.Org/epic/ staff/hoofnagle/ilbpaper.html

Dan, L. B. & Julie, E. C.(2001): Fair Use Infrastructure for Rights Management Systems. Harvard Journal of Law, Vol. 41, No. 60.

Introna, L. D. (1999): Privacy, Autonomy and Workplace Surveillance. ETHICOMP99. Available online at http://www.ccsr.cse.dmu.ac.uk/condference/abstracts99/introna.html

James, E. (2003): Securing Deliberative Autonomy, STAN. Law Review. Vol 48, No. 1

Jeff, S. (1999): Opting in, Opting Out, or No Options at All: The Fight for Control of Personal Information, 74 WASH. L, REV. 1033

John, B. (2003): A Secret War: Spike in "Spyware" Accelerates Arms Race. CNET News.com. Available online at http://news.com/2102-1023.html.

Julie, E. C. (2000): Examined Lives: Informational Privacy and the Subject as Objects. Stan. Law Rev. Vol. 52, No. 28

Julie, E. C. (2003): DRM and Privacy. Berkeley Technology Law Journal. Vol. 1, No.2

Longe O.B (2004): Software protection and Copyright issues in Contemporary Information Technology. Unpublished M.Tech Degree Thesis. Federal University of Technology, Akure, Nigeria.

Moffaert, R. (2003): "Digital Rights Management" Available online at http://www.alcatel.com/docu

Samuel, D. W. & Louis, D. B. (1990): The Right to Privacy. Harv. Law R. Vol 4, No. 193

Wurzler, R. (2004) Underwriting Hacking Insurance. Underwriting Cyber-Insurance

Available online at <http://www.pcmag.com/article2/0,1759,61635,00.asp

On Explicit Specification of Software Components with *DB-Data Enkryta*

Daramola, J.O., Bamigbola, O.M.

ABSTRACT

Although Component-based software development and component technology offer exciting and interesting potentialities for the simplification of complex software models in order to gain increased functionality, reduction in time and cost of development, the desire to enhance the user's understanding and perception of components' behaviours is still a pressing challenge. Since users do not have access to the program texts (source codes) of component developers in most cases, it becomes necessary for the developers to give a detailed specification of component functionalities, behaviour and performance in a way that will enhance their predictability even before they are deployed. In this paper we have adopted the semantics of the CSP-like WRIGHT in the specification of a visual component tool named "DB-Data Enkryta" that can be used to compress and encrypt text stored in a relational database. WRIGHT is one of the existing Architecture Description Languages (ADL) used in the formal specification of component-based systems that enables explicit specification of component and connector interfaces. This we believe is capable of giving an added insight to prospective users about the behaviour of components and enhancing interoperability among components in component based systems.

1. Introduction

Component-based software engineering affords the opportunity for independent development of components that can be used in the composition of software systems. These often include third-party components [5], [8], [21] which are selected at build-time and reconfigured for interaction at run-time. One of the major concerns of component-based development is the need to minimize instances of mismatches and misuse of components especially in the context of assembled systems [7], [9].

There is a general consensus in literature on the need to extend the level of component specification beyond the syntactic level [3], [4], [6], [11], [14], [21]. The existing mode of specification that exist in the available standard component models like CORBA, COM and JAVA are mainly syntactic, which only describes the functional properties of components. These forms of specification highlights the services provided and services required (CORBA IDL) by the components, which also represents a contract between the component and its prospective clients. These commercial Interface Definition Languages (IDLS) primarily specify the signature aspect of software components interfaces i.e. names, parameters, and data types of the provided services, they do not provide the necessary support for capturing the semantic or behavioural aspects of a component [12] including its usage, capabilities and interaction behaviour. This limited specification often leads to mismatches in behavioural interoperability between components when designing component-based systems, especially in cases where third party components are involved [12].

In order to provide a sound basis for component interoperability, accurate description, precise semantics, qualities, usage and interaction protocol between components there is a need for a more explicit specification, which entails the description of component interaction protocols and the specification of both functional and extra-functional properties.

In this paper, we have used an approach [1], [2] which is based on the WRIGHT formal specification language in the specification of a visual component called "DB-Data Enkrypta". This is a visual tool that we have constructed which can be made available as an Activex component across ActiveX platforms for compressing and encrypting a database file.

2. Motivation The motivation for this work is to ensure the precise specification of component interaction protocols in contrast to what presently exist in commercial Interface Description Languages (IDLs). In order to illustrate this, we have used the example of a

```
1. Interface Compressor_Encrypta {
2. String Compress ( in string filename, in string filedriver);
3. String Encrypta ( in string filename);
4. }

1. Interface Decompress_Decrypta {
2. String Decompress ( in string filename);
3. String Decrypta ( in string filename);
4. }
```

Figure 2.0 Overview of CORBA IDL specifications for DB-Data Enkrypta Component.

data compression and data encryption component, DB-Data-Enkrypta. This component is a composite component, which comprises of two sub-components [5]. The first sub-component is the Compressor-Encrypta component, which compresses a database file using the filename as input parameter for the Huffman compression algorithm [10]. Thereafter, it encrypts the compressed file using a data encryption standard (DES) method [13]. The second sub-component is the Decompressor-Decrypta component, which consists of methods to decrypt encrypted data and to decompress previously compressed data. The equivalent CORBA IDL definition for the DB-Data Enkrypta component is shown in figure 2.0.

From Figure 2.0, it cannot be seen in which particular order a user or a prospective client should invoke the services provided by this component in order to correctly engage it. This underscores the need for protocol specification, which will indicate the particular sequence for invoking the services provided by a component such that performance synchronization and run-time behaviour interoperability can be achieved. Considering our example, the correct sequence of invoking the operation of the DB-Data Enkrypta component is to first invoke the compressor method for compression and then the encryption method for encrypting the compressed file. We also need to first decrypt the encrypted file before we invoke the decompression method in order to get back the original file.

The configuration of this component is such that any variation from this outlined sequence of invocation will lead to system failure and yet this could not have been discovered in any way from the purely syntactic specification given in Figure 2.0. Figure 2.1 shows the collaboration diagram for these transactions.

One common approach to this kind of problem as it is being done in the specification of commercial components [15],[16],[18],[20] is the use of informal documentation, which is attached to a component explaining the protocol of its interaction in order to avoid problems. However, this kind of informal documentation leaves room for ambiguity and inconsistency and also lacks adequate basis for the

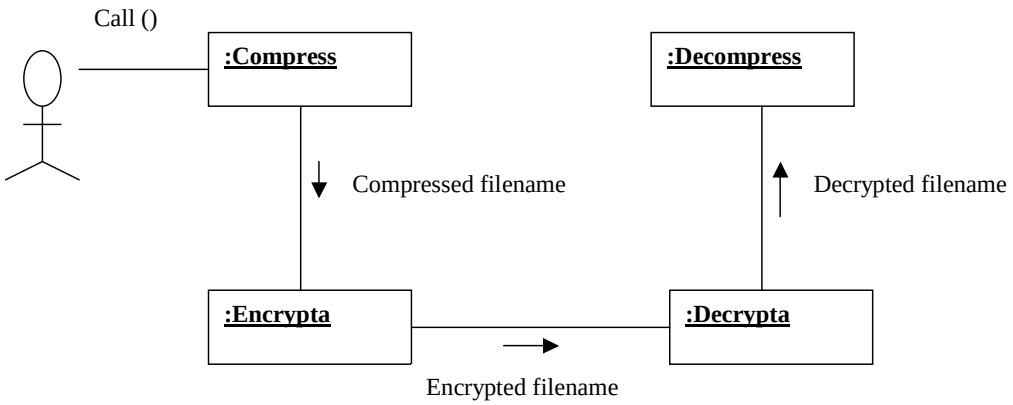


Figure 2.1 Collaboration Diagram for DB-Data Enkrypta Component [17]

precise description and analysis of the behavioural properties of a component. This makes necessary, the need to evolve a more explicit and precise way of specifying the interoperability requirements of components in a way that can be verified and proved for correctness.

Although a formal approach has been adopted, which ordinarily may not interest many developers with relatively low affinity for mathematical formalisms, we can say that the notations of WRIGHT are quite relatively mild and not difficult to understand. It became our choice language because of its attribute of containing abstraction elements for explicit specification of both components and connectors. It also has automatic support tools for checking for the completeness and consistency of specifications by using the FDR (Failure, Divergence, Refinement) as specification checker.

3. Introduction To WRIGHT Specification

WRIGHT is an Architecture Description Language that is used for the formal definition of software architecture and also provides a basis for the analysis of the system structure. The notion of WRIGHT as an architecture description language is based on the abstraction of the components in a software system as independent computational entities [5] and the connectors as a composition patterns among components [1], [19]. This makes WRIGHT very suitable for the specification of component-based systems. It also provides a precise semantics that resolves ambiguity and aids in the detection of inconsistencies. WRIGHT also provides a set of techniques that supports reasoning about system properties [1].

3.1 Key Elements of WRIGHT

The basic architectural abstractions of WRIGHT are components, connectors and configurations. It provides explicit notations for specifying each of these elements, which makes it adequate for the specification of component-based systems. Component-based systems are essentially an assembly of independent components and connectors that have been integrated for coherent behaviour in order to achieve a set of requirements. We now give a preview of each of these elements.

3.1.1 Component

A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by a third party [5].

The description of a component in WRIGHT has two parts the *Interface* and the *Computation.* An Interface consists of *Ports.* A *Port* represents an interaction in which the component may participate. The *Computation* section of a description describes what the component actually does. The *Computation* carries out the interactions described by the *Ports* and shows how they are tied together to form a coherent whole. Therefore, in WRIGHT, the *Computation* is the full specification upon which analysis of the components properties will be based.

3.1.2 Connectors

A connector encapsulates the character of the component interactions. It is the link between to components [19].

In WRIGHT connectors represents an interaction among a collection of components. The abstraction of connectors as an encapsulation of interaction among a collection of components achieves two important purposes. First is that it increases the independence of a component by structuring the way a component interacts with the rest of the system [1]. In order words, a connector provides in effect a set of requirements that a component must meet, and an information-hiding boundary that defines what expectation the component can have about its environment. The second is that it extends that applicability of analysis of components [1]. A connector between components could be as simple as a procedure call, which indicates a call-and–return pattern of control.

The WRIGHT's description of connectors divides it into a set of *Roles* and *Glue*. Each Role specifies the behaviour of a simple participant in the interaction. For example a pipe connector has two roles i.e. the source of data and the Sink (the component that receives data) while a procedure call connector has a Caller and Definer. The *Role* indicates what is expected of any component that will participate in that interaction. A *Glue* of a connector describes how the participants will work together to create an interaction. The connector *Glue* specifies how the computation of the component is composed to form a larger computation. Like the computation of a component, the *Glue* of the connector represents the full behavioural specification. It is the Glue process that will co-ordinate the components behaviour to create an interaction.

Therefore, a component specification is interpreted to mean that if the actual components obey the behaviour indicated by the roles, then the different computations of the components will be combined as indicated by the Glue [1], [2].

3.1.3 Other Elements of WRIGHT Structure

a) **Instances :** In WRIGHT, components and connectors are defined as types which are then instantiated for various uses. An instance therefore is an occurrence of a type of component or connector.

b) **Configuration:** A configuration is a collection of components' instances combined via connectors.

c) **Attachment**: This defines the topology of the configuration, by showing which components participate in which interaction. This is done by associating the ports of the components with the roles of the connectors.

d) **Styles:** This is a set of configurations types that can be adapted for an entire family of systems. Styles can also have *constraints* which represents predicates that must satisfied by any configuration declared to be a member of the style.

3.2 Some Notations of WRIGHT Specification

WRIGHT makes use of a notation that is based on CSP. CSP (Communication Specification Process) is a process specification Language for specifying patterns of behaviour and interaction. The two key features of WRIGHT behaviour specifications are Events and Processes. Some of the notations used in specifying events, and processes are shown as follows [1], [2]:

Events:

1. ¶- indicates a signalled event that terminates successfully.
2. Write - indicates a signaled event
3. b?x - indicates a process b receives data x
4. b!x - shows that a process b outputs data x, it is signaled event (initiated by the process itself).

Processes:

1. STOP – the process that does nothing.
2. Sequencing – b®P (events b occurs and then P occurs).
3. ¶®STOP – indicates a process that successfully terminates.
4. ";" - also used as sequencing operator.
5. P;Q – means P executes successfully, terminates and then Q executes and terminates.
6. b®f®§; (g®§) = b®f®g®§
7. ' - An alternative that recognizes the possibility of two behaviours in its environment. This is called the deterministic or external choice e.g. b®P ' F®Q is the process that will behave as the process P if it first observes event b and will behave as the process Q if it first observes the event F. the deterministic choice depends on what the environment does.
8. Ð – A second alternative is a process that makes an internal choice about which of two behaviour to perform. This is called non-deterministic or internal choice and uses the operator Ð. The process e ® P Ð F ® Q is the Process that will either output e and then acts as P or F and then acts as Q. The process itself decides which to do, without consulting the environment.
9. ' x: S ? P(x) – indicates an external choice between different P(x)
10. Ð x: S ? P(x) – indicates the execution of all of the different P(x) in some order.

4. Eplicit Specification with DB-Data Enkryta

The DB-Data Enkrypta is a composite component composed of two sub-components, the Compressor and Decomprocessor with two ports each. The Compression–Encryption service provided by the DB-Data Enckrypta entails the compression of a database file specified by its filename using a Huffman encoding algorithm [Huffman]. The compressed output is then passed to the encryption method of the Compressor sub-component to encrypt the data using a DES (Data Encryption Standard) algorithm. The Decompressor–Decryption service involves the decryption of encrypted data with the decryption method and then the decompression of decrypted data to get the original data. Figure 4.0 shows the internal (White-box) view of the DB-Enkrypta component.

It is composed of two subcomponents as shown. The external ports are

connected to the ports of the contained subcomponents by delegation connectors [11]. The direction of the delegation connectors indicates that the input ports *in1* and *in2* ports are associated with required interfaces and the output ports out1 and out2 are associated with

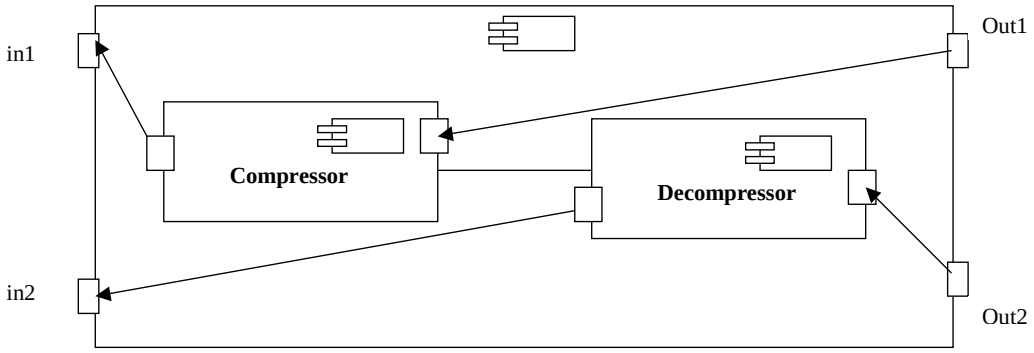


Figure 4.0 White-box view of the DB-Enkrypta component-based system [11]

the provided interfaces. The internal subcomponents are connected by an unlabeled assembly connector [11].

4.1 Formal Specification of DB-Data Enkrypta with WRIGHT

Component *Compressor*(string:x:1..2)

Port Compress = read?x ® compressá$_x$ñ ' close® §

Port Encrypta = encryptaá$_x$ñ ® write!x ® Ð close ® §

Computation = compress.read?x ® compress.compressá$_x$ñ ®

encryptaá$_x$ñ ® encrypta.write!x ® **computation**

' (compress.close ® encrypta.close ® §)

Component *Decompressor*(string:Cipherx)

Port decrypta = read?Cipherx ® decryptaá$_{Cipherx}$ñ ' close® §

Port decompress = decompressá$_{Cipherx}$ñ ® write!x ® decompress Ð close ® §

Computation = decrypta.read?cipherx ® decrypta.decryptá$_{Cipherx}$ñ ®

decompress.decompressá$_x$ñ ® decompress.write!x ®

computation ' (decrypta.close ® decompress.close ® §)

Connector call_Compressor

Role caller = call ® return ® caller Ð §

Role compressor = call ® return ® compressor ' §

Glue = caller.call ® compressor.call ® **Glue**

' compressor.return ® caller.return ® **Glue** ' §

Connector call_decompressor

Role caller = call ® return ® caller Ð §

Role decompressor = call ® return ® decompressor ' §

Glue = caller.call ® decompressor.call ® **Glue**

' decompressor.return ® caller.return ® **Glue** ' §

5.0 Component Specification

The approach of using WRIGHT specification effectively addresses the issue of protocol specification, affording the opportunity to clearly spell out in a formal way the order of component interaction as demonstrated by our example. This is an improvement on the existing standard especially the Interface Definition Languages (IDL) that is mostly used in the specification of commercial components [15], [16], [20]. The specifications are essentially syntactic, and say little about the other aspects of a component that a prospective user will be interested in. A detailed explicit specification should therefore include the specification of component at the syntactic, semantic, protocol and extra-functional levels.

One of the key advantages of WRIGHT specification as shown with our example is that it has adequate abstraction elements to explicitly specify a component at the syntactic, semantic and protocol level. The syntactic aspect deals with the names, parameters, and

data types of the provided services, the semantic aspect deals with the specification of constraints that determines the conditions for components interaction, [17]. The protocol aspect indicates the order of interaction while the extra-functional aspect deals with properties like the performance and reliability rating of the component [23], [24].

In our future work, we intend to extend our explicit specification to include the extra-functional properties like reliability using the *CoRe* (Component Reliability) method [14] and performance using a performance engineering approach [22]. This we believe, will further boost user perception of such component and enhance interoperability and minimise component mismatches.

6.0 Conclusion

We have shown with our example that explicit specification of components which includes syntactic, semantic and protocol specification is possible with an ADL like WRIGHT. This is recommended for component developers as it has the tendency to greatly improve the understanding and perception of component behaviour, boost component interoperability in order to minimise misuse and mismatches in the composition of component-based system.

References

1. Allen, R.J. (1997): A Formal Approach to Software Architecture. Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh.

2. Allen, R and Garlan, D. (1997): A Formal basis for architectural connection; ACM Transaction on software Engineering and methodology; 6(3); 213-249.

3. Borgida A., and Devanbu, P. (1999): Adding more "DL" to IDL: Towards more knowledgeable component interoperability; In proc. 21st Int'L Conference on Software Engineering (ICSE), Los Angeles, CA. ,378-387.

4. Canal, C. and Pimentol, E., Troya, J.M., Vallecillo, A., (2000): Extending CORBA interfaces with Protocols; The Computer Journal 44(5), 448-462.

5. Crnkovic, I. Hnich, B., Jonsson, T., and Kiziltan, Z. (2002): Specification, Implementation and Deployment of Computers; Communications of the ACM, Vol. 45. No. 10. 35-40.

6. Han, J. (1998): A Comprehensive Interface Definition Framework for Software Components; In proc. Asia-Pacific Software Engineering Conference (APSEC), Taipei, Taiwan. 110-117.

7. Han, J. and Ker, K.K., (2003): Ensuring Compatible Interactions with Component-based Software System; In proc. Asia-Pacific software Engineering conference (APSEC), Chiangmai, Thailand; IEEE Computer Society. 436-445.

8. Han, J. [1999]; An approach to software component specification; In Proceedings of Internal Workshop on Component-Based Software Engineering; Los Angeles, U.S.A.

9. http://www.cbs.colognet.org/Sepecifiaction.php: Component Specification and Retrieval

10. http://www.crypto.com: Huffman Encoding Algorithms

11. Ivers, J. Clements, P., Garden, D, Nord, R., Schmerl, R. and Silva, I.O.R. (2004): Documenting component and connector views with UML 2.0; Technical Report CMU/SEI-2004-TR-008 ESC-TR-2004-008; Carnegie Mellon Institute, Pittsburgh.

12. Jin, Y., and Han, J. (2004): PEIDL: An Interaction Protocol Specification Language for Software Components; Center for Component and Enterprise System, School of Information Technology, Swinburne University of Technology; Technical Report: SUTIT-FR2004. 02/SUT-Cecses-tr002.

13. Krammer, R.G. (1999): Data Encryption Standard (DES); Federal Information Processing Standard Publications, FIPS PUB 46-3, U.S.A.

14. McGregor, J.D., Stafford, J.A., Cho, I. (2003): "Measuring and Communicating Component Reliability, Software Engineering Research and Applications *(*SERA*)*, San Francisco, CA, USA, Selected Revised Papers Series: Lecture Notes in Computer Science, Vol. 3026 Ramamoorthy, C.V.; Lee, Roger Y.; Lee, Kyung Whan (Eds.)2004, pp. 74—86.

15. Microsoft (1996), The Component Object Model Specification; Report V 0.99, Microsoft Standards, Redmond WA.

16. OMG orbos/99-04-17 (2000); CORBA Component Model Volume 1.

17. OMG: OMG-Unified Modelling Language Report V1.5; Object Management Group; http//www.omg.org/

18. OMG (2000): The Common Object Request Broker: Architecture and Specification, Report V.25, OMG Standard Collections.

19. Plasil, F. and Wisnovsky, W. (2002): Behaviour Protocols of Software Components; IEEE Transaction on software Engineering 28(11); 1056 – 1076.

20. Sun Microsystems: Javabeans 1.0 Specification; http://www.java.sun.com/beans

21. Zaremski A.M., Wing, J.M. (1995): Specification Matching of Software Components; In proc. 3rd ACM SIGSOFT Symposium on the Foundation of Software Engineering. 6-17.

22. Wu,X., McMullan, D., Woodside, M. : Component Based Performance Prediction; http//www.csse.monash.edu.au/~hws/cgi-bin/CBSE6/Proceedings/papersfinal/p24.pdf

23. Zschlaer, S. : Formal specification of Non-Functional properties of Component-Based Systems; http//www.comquad.inf.tu-dresden.de/nfc04/papers/nfc04_3.pdf

24. Zschlaer, S. (2004): Towards a semantic Framework for Non-Functional specification of component-based system. In stainmetz, R., Manthe, A., eds: Proc. IEE Computer society.

Software Engineers and ICT Development in Nigeria: The Challenges, Opportunities and Potentials

Nnebe S.E., Kenneth Agbasi

ABSTRACT

Since the building of information systems, which is about the life wire of information and communication technologies (ICT) falls within the domain of software engineering. Therefore, the rapidly increasing production of information in society makes a number of demands on the personal qualifications of the software engineers. In the information society, the ICT qualification of software engineers such as processing and handling of information and the role as an organizer and constructor of a number of multiple version software has become as important as the traditional basic qualification: reading, writing, and arithmetic. This paper outlines the challenges, opportunities, and the potentials of software engineers in ICT development in Nigeria. It also presents the problems facing the continuing development of ICT as an indispensable tool for socio-economic development in the country. Greater priority on developing this aspect of human capital. therefore, will facilitate socio-economic development and as such, bridge the digital divide.

INTRODUCTION

Computer and communications technologies, the two elements of the Information and Communication Technology (ICT) are transforming national and global societies and economies into information-driven societies and economies. Presently ICT account for more than 5 percent of the global GDP, and much more in industrialized countries. ICT represents a revolution as important as it was the Industrial Revolution. It is already crossing national and regional boundaries creating challenges for slow-moving bureaucracies, both public and private, and changing the way we act and how we think. It is generating a change that will be difficult to manage but impossible to resist. ICT carries on high promise both in human and economic terms. Benefits could be obtained among others in: education at all levels, job training, health care, food security, environment management, and government efficiency but it is necessary to develop and adapt new systems and technologies to make a suitable use of ICT.

Governments and people around the world have started appreciating the ability of Information and Communications Technology (ICT) to stimulate rapid development in all sectors of the economy. ICT is redefining the way we do almost everything and it is a ready tool for all strata of society- it is as much a tool to the President of any nation in governance as it is a tool for the housewife in her daily chores! (Ajayi, 2003). Thankfully, Nigeria is exploring the benefits of ICT as well.

In Nigeria, the ICT Revolution started after the return to democratic rule in 1999. The country had gone through an extended period of military dictatorship prior to this time. This came with an attendant apathy for the development of a platform for developing ICT in the country. In some cases, it was even believed that ICT would pose a security threat! It dawned on the government that the digital divide would only continue to widen except the issue of developing ICT in the country was given the priority attention it deserved.

This transformation, rapid spread of computers and information technology and the necessity to develop and adapt new systems and technologies has generated a need for highly trained workers to design and develop new hardware and software systems and to incorporate new technologies. These workers-computer systems analysts, database administrators, and computer scientists-include a wide range of computer specialists. Job tasks and occupational titles used to describe these workers evolve rapidly, reflecting new areas of specialization or changes in technology, as well as the preferences and practices of employers. For instance, the growth of the Internet and the expansion of the World Wide Web (the graphical portion of the Internet) have generated a variety of occupations related to the design, development, and maintenance of Web sites and their servers. For example, *webmasters* are responsible for all technical aspects of a Web site, including performance issues such as speed of access, and for approving the content of the site. Internet

developers or Web developers, also called Web designers, are responsible for day-to-day site design and creation.

Amidst the seeming sectoral responses to technological advancement in Nigeria, the political environment and the enabling regulatory environment for which ICT can thrive has changed rather slowly. Aware of the role and relevance of software as ICT tool in fighting global poverty, enhancing human development, facilitating communications, creating wealth and relationship management, the global push for ICT adoption and adaptation at all levels is of course consequential.

SOFTWARE ENGINEERING TECHNOLOGY

Technological trends have played significant roles in the evolution of the field of software engineering. The most important influence has been that of the change in the balance of hardware and software cost. Where the cost of a computerized system used to be determined largely by hardware costs, the software was an insignificant factor. Today the software component can account for far more than half of a system cost. The declining cost of hardware and rising cost of software have tipped the balance further in the direction of software, setting in motion a new economical emphasis in the development of software engineering.

The birth and evolution of software engineering proper as a discipline within Computer science can be traced to the evaluating and maturing view of programming activity. As computer became cheaper and more common, more people using them, higher-level languages were invented to make communication with the machine easier. In the middle of the 1960s truly large software systems were attempted commercially. The large projects were the source of the realizations that building large software systems were materially different from building smaller ones.

Another evolutionary trend has been internal to the field itself. There has been a growing emphasis on viewing software engineering as dealing with more than just “coding” instead the software is viewed as having an entire life cycle, starting from conception, and continuing through design, development, and maintenance. This shift of emphasis away from coding has sparked the development of methodologies and sophisticated tools to support teams involved and test plans, in the entire software life cycle (Panzi, 1981).

Information and Communication Technology and its applications are developing exponentially. The global ICT markets are growing and expanding. It has become largest industry in the world, larger than textiles, automobiles and construction. The information technology applications have revolutionized the organizational program for production of systems and services. The rapid decline in relative price of information handling brought about through microelectronics and digital communications has fostered information intensity in industry product and processes rather than energy and material intensity.

Based on the rapid impact on ICT human resources development, today, all the industrialized countries and an increased number of newly industrialized countries use ICT in areas as diverse as macroeconomic planning and decision making, public administration, education, healthcare, manufacturing, finance and banking, transport, commerce, publishing, energy conservation and management. These afore mentioned areas of human endeavour involve information systems, which depend on software resources to help end users use computer hardware to transform data resources into variety of information products. Software is needed to accomplish the input, processing, output, and storage and control activities of these information systems. (Wendy, 1997).

As areas of ICT human resources development increases, software engineering is committed to develop the full potential of the ICT manpower, to arm them with the right skills and to perform in the most efficient and cost-effective way. Software engineers intend to use latest technologies to change the economics and logistics of learning radically in order to yield dramatic benefits in information systems in terms of cost, accessibility, convenience and effectiveness. These could be better achieved committing themselves to certain ethics and principles.

SOFTWARE ENGINEERING CODE OF ETHICS AND PROFESSIONAL PRACTICES

That computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large is now a widely accepted fact. Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems that drive and coordinate the hardware components of these computers. Because of their roles in developing these software systems, software engineers have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence others to do good or cause harm.

To ensure, as much as possible, that their efforts will be used for good, software engineers must commit

themselves to making software engineering a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers in Nigeria shall adhere to the following Code of Ethics and Professional Practice as recommended by the IEEE-CS/ACM joint task force;

1. PUBLIC <http://www.computer.org/tab/seprof/code.htm> - Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER <http://www.computer.org/tab/seprof/code.htm> - Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest.
3. PRODUCT <http://www.computer.org/tab/seprof/code.htm> - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT <http://www.computer.org/tab/seprof/code.htm> - Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT <http://www.computer.org/tab/seprof/code.htm>- Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION <http://www.computer.org/tab/seprof/code.htm> - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES <http://www.computer.org/tab/seprof/code.htm> - Software engineers shall be fair to and supportive of their colleagues.
8. SELF <http://www.computer.org/tab/seprof/code.htm> - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

The Code contains eight Principles related to the behaviour of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession. The Principles identify the ethically responsible relationships in which individuals, groups, and organizations participate and the primary obligations within these relationships. The Clauses of each Principle are illustrations of some of the obligations included in these relationships. These obligations are founded in the software engineer's humanity, in special care owed to people affected by the work of software engineers, and in the unique elements of the practice of software engineering. The Code prescribes these as obligations of anyone claiming to be or aspiring to be a software engineer.

These Principles should influence software engineers to consider broadly who is affected by their work; to examine if they and their colleagues are treating other human beings with due respect; to consider how the public, if reasonably well informed, would view their decisions; to analyze how the least empowered will be affected by their decisions; and to consider whether their acts would be judged worthy of the ideal professional working as a software engineer. In all these judgments concern for the health, safety and welfare of the public is primary; that is, the "Public Interest" is central to this Code.

SOFTWARE ENGINEERING AND ICT DEVELOPMENT IN NIGERIA

As we try to preach the benefits of information revolution to Nigerians, we need trained personnel (the software engineers) to design and implement networks that are robust and cost effective. While 'the digital divide' conventionally refers to inequality of access to ICT services such as telephone, computer and Internet, 'the knowledge divide', refers to the inequality in the capability and skills to generate and use knowledge. (Ndukwe 2004).

In several aspects of ICT, satisfactory solutions to problems particular to developing nations have not yet been found. Nigeria must therefore ensure a well organized, human resources development approach in this vital sector such that professional education and training in areas of software engineering in our universities and other institutions must be well adapted to a well-articulated set of objectives for the nation.

While progress has been made in the past decade, the narrowing of the gap in terms of ICT penetration indications are such that developing and poor nations may in fact be lagging further behind, in such areas as software engineering and knowledge creation. Nigeria must therefore, expand and modernize educational facilities in order to facilitate the creation of an all inclusive knowledge base.

It is important that the Nigerian nation continues to accord priority to the development of software engineers with a view to developing necessary infrastructures and access to ICT's for its citizens. Sustained policies aimed at encouraging widespread availability of these essential infrastructures must be

placed at the front burner just as is the case with the more developed nations of the world, which have continued to expand their knowledge base and as such upgrade their ICT resources.

While much research on software engineering is taking place in the developed world, very little of it is happening here. Critical to this relative low level of engagement with the sector in Nigeria in particular is the poor state of ICT accessibility and software engineering technology in most of the academic institutions of the country. There is the need to strengthen interest in these two sectors with the hope that the institutions and their researchers would be empowered to utilize them both as tools for research and administration, as well as subject of research.

To create an enabling environment for the use of ICT, foster information exchange among local scientists, and to facilitate the interactions and collaboration between researchers in institutions and the world, several initiatives have been made by the government of Nigeria and other agencies to develop the ICT-infrastructure so as to bridge the digital divide between Nigeria and the world.

Investment in software engineering is critical to narrowing the knowledge gap and is fundamental to the development of the capacity for integrating knowledge into social and economic activities and for participating in today's digital economy.

ICT INFRASTRUCTURAL DEVELOPMENT IN NIGERIA (DEC 1999 - DEC 2004)

The development of a modern nation to its full potential today can never be attained without adequate ICT infrastructure for adequate ICT infrastructure utilization means adequate software empowerment. In today's world, modern digital telecommunications networks are as necessary to economic growth- and to attracting foreign investment as are programs dedicated to promoting healthcare, electricity, transportation and agriculture. It is also true that reliable telecommunications networks can improve the productivity and efficiency of other sectors of the economy and enhance the quality of life generally.

Regrettably however, most of the values derivable from infocommunications development have been concentrated in the developed countries of the world. Africa for instance has less than 3% of the world's main lines although it accounts for more than 12% of the world's population. (Ndukwe, 2004).

In Nigeria the telephone density is estimated at about 5 telephones for about 100 people or 5%. As telephones tend to be concentrated in the cities, access in rural areas is even much more limited and non-existent in many parts of the country.

Studies by the International Telecommunications Union (ITU) and the World Bank show that there is a direct correlation between telephone penetration and economic growth.

Information tools such as telephones, personal computers, and the Internet are increasingly critical to economic success and personal advancement. There is therefore a big divide separating the world's information "haves" and "have nots" which is what is now commonly referred to as the digital divide - the divide between those with access to new info-communications technologies and those without.

Since year 2000, the Nigerian Communications Commission (NCC) has licensed Digital mobile operators, Fixed wireless Access Operators, two Long Distance Operators, Internet Service Providers and a Second National Carrier, thus ensuring competition in all segments of the market. This activity has increased and promoted rapid deployment of ICT services, resulting in exponential growth in the number of telephone lines. It is instructive to note that while connected lines only grew at an average of 10,000 lines per annum in the four decades between independence in 1960 and end of 2000, in the last three years, an average growth rate of 2 million lines per annum was attained. Table 1 below summarizes the ICT infrastructural development in Nigeria (Dec. 1999-Dec. 2004). NCC (2004).

Infrastructure	December 1999	December 2002	December 2003	December 2004
Number of Connected Fixed Lines	450,000	702,000	850,000	1,120,000
Number of Connected Digital Mobile Lines	None	1,594,179	3,100,000	9,200,000
Number of National Carriers	1	2	2	2
Number of Operating ISPs	18	30	35	40
Number of Active Licensed Fixed Line Operators	9	16	30	17
Number of Licensed Mobile Operators	1	4	4	4
Private Investment	$50m USD	$2,100m USD	$4,000m USD (est.)	$6,000m USD (est.)

Table 1: ICT Infrastructural development In Nigeria (Dec. 2000-Dec. 2004). Source: NCC (2004).

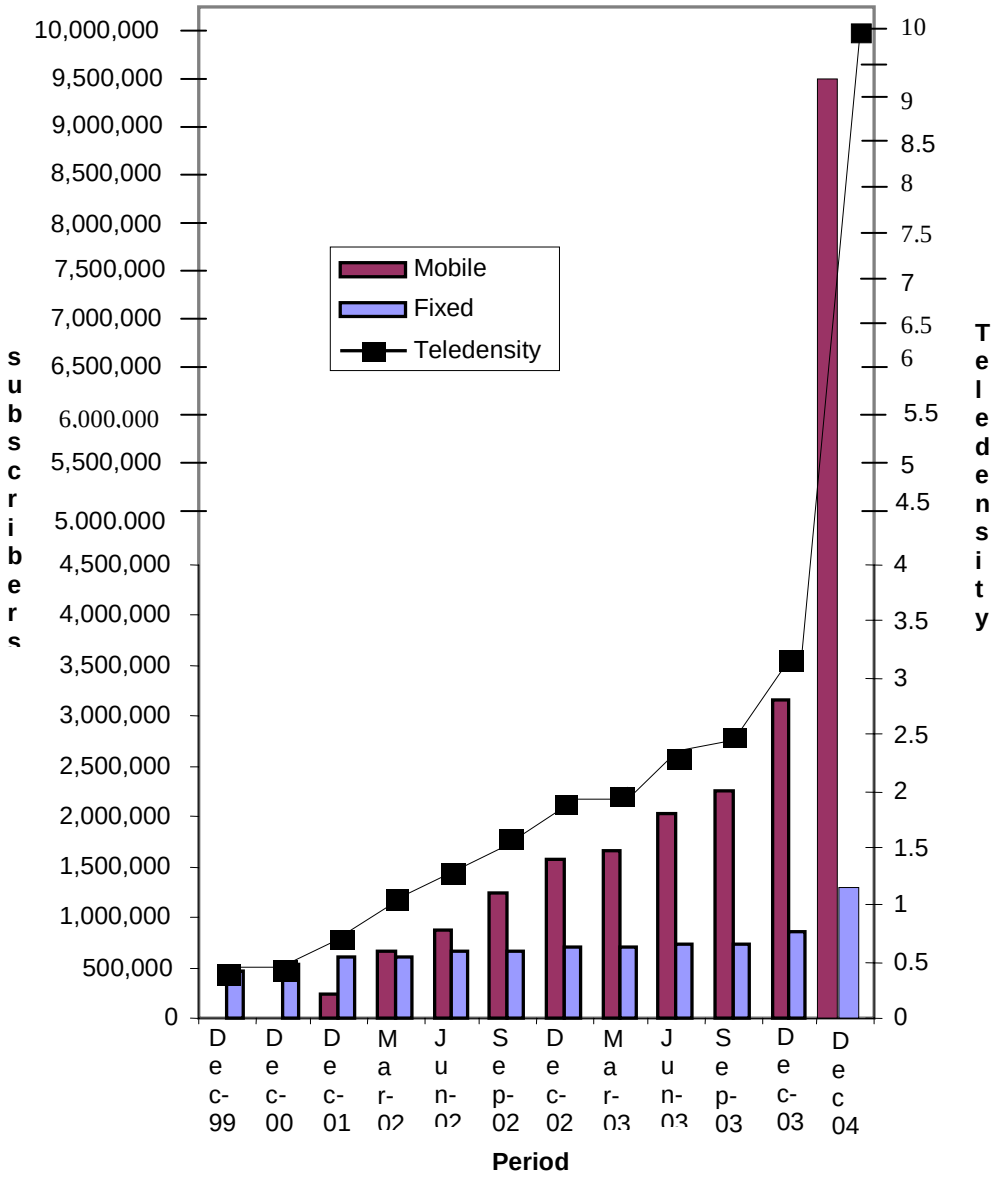


Fig 1: Teledensity as at December 2004. Source: NCC (2004).

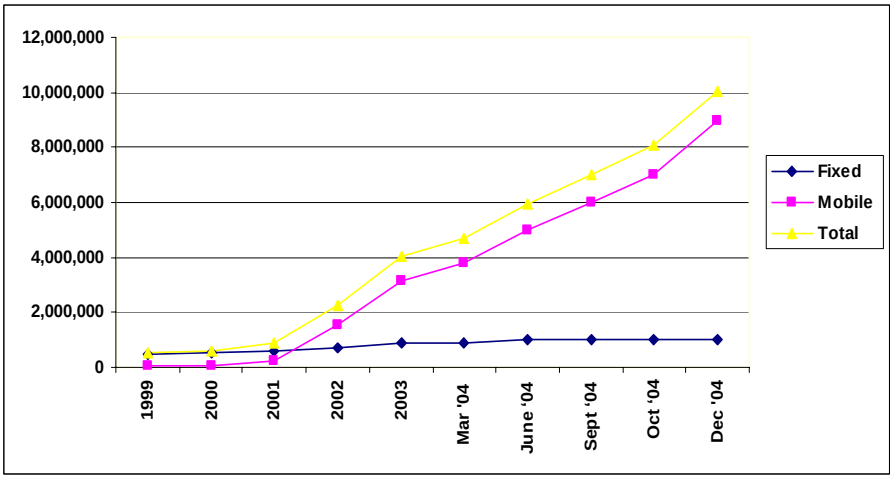


Fig 2: Growth of subscribers in the Nigerian network - Source: NCC

Total teledensity was 0.40 lines per 100 inhabitants in 1999; reached 1.96 in December 2002, It increased to 3.33 in December 2003. By December 2004, the total teledensity was estimated at 10.00 lines per 100 inhabitants

As of June 2004, Nigeria had attained over 6 million lines, (5 million of which are digital mobile lines). Total teledensity, which was just 0.4 lines per 100 inhabitants in 1999 stood at 10.0 per 100 inhabitants by end of December, 2004.

CHALLENGES, OPPORTUNITIES AND POTENTIALS OF SOFTWARE ENGINEERS FOR ICT DEVELOPMENT IN NIGERIA

The major deficiencies in software engineering services on the African continent can be traced to neglect of the past and the lack of adequate infrastructures and facilities, resulting from long years of state ownership and control of economic activities. However, following global trends in the last two decades, African countries have created, and are adopting national policies that are private sector driven, and aimed at delivering services to communities throughout the countries. In the ICT environment, these policies have made some positive impact on the human and economic development of the nation. (Ndukwe, 2004). In Nigeria, the story is now more exciting than was the case before the dawn of the new millennium. Though the country still has a long way to go as far as development of software engineering technology and ICT infrastructure is concerned, there has been remarkable progress in the installation and uptake of subscriber lines in the last four years (see fig 1 above). However a lot more could have been achieved if not for a number of factors including, the lack of adequate infrastructures and appropriate technologies. Consequently, the challenges to achieving increasing access to ICT infrastructure in the country will include ensuring deployment of appropriate infrastructures, and adequate connectivity within states and across the country.

The major challenge in the development of the full potential of the current generation of software engineering in ICT for education, research and development in the country is brain drain, which has resulted in the lack of the critical mass of ICT-engineers and scientists relevant for undertaking ICT-related projects professionally. They are faced with the challenges to build systems that enable and support the kinds of business initiatives typical of the internet age, combining innovation with quality, scalability, and robustness a very tall order indeed. This requires approaches that;

(i) Exploit new technologies to their full potential

(ii) Rive new business initiatives to provide competitive advantage in increasingly crowded market place of solutions.

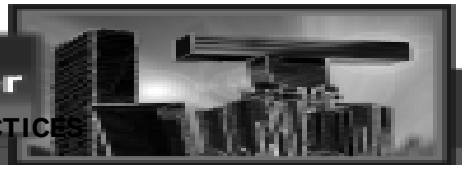
Meeting these needs is always the challenge faced by enterprise application development in the Internet age.

Another major obstacle is the lack of an enabling environment and a sound ICT-roadmap and strategies by policy makers resulting in uncoordinated and unsustainable ICT-development activities. Other challenges include; identification of information sources that meet the needs of users, poor quality of service of the Internet and telecommunication services, ineffective management of network traffic and infrastructure and unreliable power supply.

In today's information and knowledge driven world, a whole new concept of skills is required. A relevant education in software engineering is more important today than ever, because today's networked world demands a work force that understands how to use technology as a tool to increase productivity and creativity. These skills include "information reasoning", a process in which reliable sources of information are identified, effectively accessed, understood, contexualized and communicated to the society we live in. The professional development of software engineers sits at the heart of any successful technology and education programme. They need not only formal training, but also sustained and ongoing support from their colleagues to help them learn how best to integrate technology into their activities. Training must go well beyond basic cutting-and-pasting.

In the light of the above synopsis, government should make more powerful pronouncements on Nigeria's digital revolution. All schools in Nigeria should be equipped with IT laboratories stuffed with software kits. Every secondary school in Nigeria should be connected to the Internet in order to spread and propagate IT awareness among our youths. Government and private sectors should support engineering programmes adequately in tertiary institutions (for instance, India graduates about 178,000 software engineers yearly, second only to the United States). (OVIA, 2001). University lecturers should be exposed to more IT conferences and seminars both locally and internationally. They should create awareness on the importance of IT as a way of life, and a part of our business culture. Import duties on computers and IT equipment should be reduced to zero. Digital villages (Nigeria's belated Silicon Valley) where software developers would test out their knowledge and skills should also be created.

Quoting an excerpt from the Silicon Boys and their valley dream by David Kaplan: " if the Silicon Valley were a



nation, it would rank among the world 12 largest economies". The boys "love to live in the valley so they can work there." Nigeria should learn to patronize indigenous software. Government should make research funds available. For Nigeria's digital revolution to blossom, government must recognize that there is a direct relationship between a highly digitalized economy and the prosperity of that economy. Government should demonstrate serious commitment in creating digital villages in strategic business districts, where adequate infrastructure and security are provided for software engineers.

Consequently, the solution strategy towards bridging the digital divide demands an aggressive human capacity building in ICT through software training workshops, seminars, and courses in collaboration with local and international institutions. In fact, an integral part of improvement in Nigeria in the 21st century will be a deployment of software engineering technology, to take the advantage of all the possibilities of the latest technology, and if we take advantage of the opportunities presented to us, ICT will enhance socio-economic development in all areas of our life.

CONCLUSION

As our society is growing increasingly dependent on software for ICTs, so also is the task and challenges of software engineers in Nigeria. The critical functions from industry to education's software provide the glue for many services and devices on which we now rely our banking services, home security devices, air traffic control, air plane navigation, etc. As software is being used to provide more functions, our need for larger and more complex ones grows. This growth of society reliance on software both promise exciting opportunities and potentials for the software engineering community and at the same time places responsibility on the community to find better ways of meeting society's expectations. To meet these expectations the software at a reasonable cost and within a reasonable schedule is targeted. That is exactly the role and potential of software engineering in Nigeria.

REFERENCES

Ajayi G. O (2003); "NITDA and ICT in Nigeria", The National Information Technology Development Agency (NITDA), Abuja, Nigeria

Hiroshi, K. (2002); "The changing role of the Dag Hammarskjold library - Bridging the gap between developing and developed countries", New York 1.

Intsiful, J, Okyere, P.F, Shiloh, O. (2003); "Use of ICT for education, research and development in Ghana: challenges, opportunities and potentials", 2003 Round Table on Developing Countries Access to Scientific Knowledge, the Abdus Salam ICTP, Trieste, Italy.

ITU, NW(Network Wizards) internet host surveys and UNDP human development report 2004.

Kozma, R. (1999); "World Links for Development: Accomplishments and challenges". Monitoring and evaluation annual report 1998 - 1999 SRI international,

Ndukwe, E.C. (2004); "The imperative of accelerating the deployment of Information and Communications Technologies (ICTS) for social and economic development", paper presented at the 11th Herbert Macaulay Memorial Lecture, UNN.

Ndukwe, C. A. (2004); "Information and Communications Technology, Science and Medicine in the 21st century Nigeria" A landmark public lecture event held by the College of Medicine University of Nigeria Enugu Campus Rotary hall

Ndukwe, E.C. (2003); **"**Building consumer confidence in ICTs", International Telecommunication Union (ITU) WORLD 2003, Geneva

Nnebe S.E. Momodu, I.B.A, Okoro, F.M (2004); " Information and Communication Technologies in education in developing nations: The Nigerian perspective", Advances in Natural and applied Sciences research vol. 2 (1) pp26-35.

Ovia J. (2001); "Financing Nigeria's digital revolution via SMEs" CTO 2001 seminar organised by the US Embassy, Lagos

Panzi, D. J (1981); "Methods for evaluating software development techniques", Journal of systems and software, vol 2, pp. 133-137.

Parnas B.A, (1987); "Comparative study of programming languages". New York American Elsevier,

Sandro M. R. (2003); "ICT Development: The Experience of the Abdus Salam ICTP", *2003 Round Table on Developing Countries Access to Scientific Knowledge, The Abdus Salam ICTP, Trieste, Italy*

Wendy R.(1997); "Strategic management information system" (2nd Edition) Financial times (pitman publishing)

Software Freedom as the Future of Nigeria's Economic Growth

Adesesan Barnabas Adeyemo; Oladipo, Onaolapo Francisca

ABSTRACT

This paper underlines the role of software freedom in achieving Nigeria's economic growth. We present an analysis of Open Source Software/Free Software (OSS/FS) and analyze reasons why software should be free. We specially emphasize the concepts of Software Freedom and finally argue that free software could turn from a fad into an efficient economic institution under certain conditions. i.e. if developer communities could be created with appropriate incentive structures and if sufficient initial momentum could foster its diffusion relying on purely cultural feelings such as giving "freely" to the community in the hopes that the community will be better off for it in the end. The idea behind this work stems from the fact that digital information technology contributes to a nations economic growth by making it easier to copy and modify information and computers promise to make this easier for all of us. In addition, we believe that to achieve proper economic growth, the society needs information that is truly available to its citizens; for example, programs that people can read, fix, adapt, and improve, not just operate as is obtained in the typical black box delivered by software owners that we could not study or change. We show that software freedom is necessary for a good way of life, and permit useful programs to foster a community of goodwill, cooperation, and collaboration.

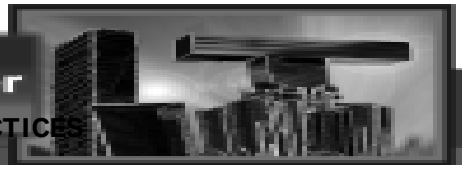
Introduction

One major need of a nation proposing to achieve Economic Growth is information that is truly available to its citizens [11]. These includes, software products that people can read, fix, adapt, and improve, not just operate. Proprietary software vendors on the other hand typically deliver a black box that we cannot study or change. To achieve economic growth, we also need freedom to control part of our own lives, such as software freedom, and we lose this freedom when a program has an owner. Above all the Nigerian society needs to encourage the spirit of voluntary cooperation in its citizens. When software owners tell us that helping our neighbors in a natural way is "piracy", they pollute our civic spirit necessary to achieve the level of economic growth we desire. This paper analyses the concept of Open Source Software/Free Software (OSS/FS) and present its importance towards achieving Nigeria's economic growth. The paper gives reasons why software should be free and specifically examine the concepts of Rights, Duty, and Making a Living in a software community. Business models that can assist in achieving economic growth in the face of free software are examined. In conclusion, the idea of creating value from Free Software in a developing country with special reference to today's Nigeria was discussed.

The Concept of Software freedom

The term software, refers to both the operating systems, and the applications, such as electronic mail and other communications, spreadsheets, electronic commerce, writing tools, sending and receiving FAXes, Web site creation, engineering, research, mathematical computations, modeling, image manipulation, and networking. It also refers to applications that are embedded in a machine, applications that control a fuel injector, or operate a telephone, or control a washing machine[1].

There are many shared principles between OSS/FS and society building especially the free and equal access to information[14].The concept of Open Source/ Free Software is about communities, communities that have been easier to create with the advent of globally networked computers. It is about "scratching an itch" to solve a problem, but it is also about giving "freely" to the community in the hopes that the community will be better off for it in the end [13]. The term "free" in free software should be equated with freedom, and as such people who use "free" software should be: Free to run the software for any purpose, free to modify the software to suit their needs, free to redistribute of the software gratis or for a fee, free to distribute modified versions of the software. The term "free" should be equated with liberation, and not necessarily giving



without return made or expected. The term "open source" is used to describe how software is licensed.

The opposite of OSS/FS is "closed" or "proprietary" software. Software for which the source code can be viewed, but cannot be modified and redistributed without further limitation are not considered here since they do not meet the definition of OSS/FS. Many OSS/FS programs are commercial programs, thus OSS/FS is not equivalent to "non-commercial" software. OSS/FS programs are not in the "public domain" (which has a specific legal meaning). OSS/FS is not "freeware"; freeware is usually defined as proprietary software given away without cost, and does not provide any right to examine, modify, or redistribute the source code. Free software is software that can be copied, studied, modified and redistributed. It is straightforward to copy and distribute software that is hard to study and modify. This is often done with Microsoft products, even though such actions are banned. Not much can be gained from software that is hard to study and hard to modify since the user learns nothing from it and cannot improve it. In the short run, the user benefits because such software makes it possible for the user to run his computer; which is why most people use such software. But in the long run, the user loses also since he can neither learn from it or advance.

Why Software Should be free

There are two reasons to prefer free software to restricted-distribution software. The first and most important reason is that a free society is better than the alternative. Open Source Software gives software users the freedom to create and use software. This is part of living in an economically viable society. The citizens have the legal right to choose a business, to choose a vendor, to choose software, to share with others, and to collaborate. A second reason is that free software, over time, tends to become more reliable, efficient, and secure. As a practical matter, the key to the good use of software is to ensure freedom. In software, this leads to reliability, efficiency, and security, to lower prices, to collaboration, and fewer barriers to entry and use as more people will be able to enter into the software industry by climbing on the shoulders of early developers.

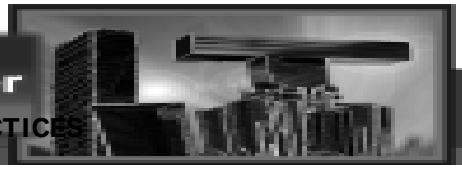
The implementation of OSS/FS principles in communities represents a method for building the community and foster collaboration. As people add what they can to the community, the community is strengthened. The rewards for these contributions are rarely monetary. Instead, the contributions are paid for with respect. People who give freely of themselves and their time are rewarded by the community as experts whose opinions are to be taken seriously. True, participation in open source software activities does not always put food on the table, but neither do other community-based activities our society values to one degree or another such as participation in community clean-ups, being involved in church activities, picking up litter, giving directions to a stranger, supporting charities, participating in fund-raisers, etc.

The right to copy, study, modify, and redistribute

The key, is freedom, the legal right to copy, study, modify, and redistribute software[1[, [11], [12]. Rights generate freedom and it is important to ensure all these rights in the Nigerian Software Industry to achieve the nations economic growth. Without them, the small time programmers and the new entrants into the software industry lose the social and technical benefits.

The right to copy: Everyone with access to a computer owns or manages a software factory, a device for 'manufacturing' software, that is to say, for making new copies. Because copying software is so easy, we do not use the word `manufacturing'; we usually do not even think of it as a kind of manufacturing, but it is. The right to copy software is the right to use your property, your Computer system, your own means of production.

The right to study: This right is of little direct interest to people who are not programmers. It is like the right of a doctor to study medicine or lawyer to read legal textbooks. Unless you are in the profession, you probably wish to avoid such study. However, this right to study has several implications, both for those who program and for everyone else. The right to study means that people in places like Mexico, or Germany, or China or Nigeria, can study the same code as people in Japan or the United States. It means that these people are not prevented from learning how others succeeded. Many programmers work under restrictions that forbid them from seeing others' code. All they see are the toy programs of school textbooks. Many years ago, a wise man said that the best way to see ahead and to advance is to sit on the shoulders of a giant. Programmers who are unable to see others' code do not sit on the shoulders of anyone; they are thrown into the mud. The right to study is the right to look ahead, the right to advance. Moreover, the right to study means that the software itself must be made available in a manner that humans can read. Software comes in two forms, one readable only by computers and the other readable only by people. The form that a computer can read is what the computer runs. This form is called a binary or executable. The form that a human can read is called source code. It is what a



human programmer creates, and is translated by another computer program into the binary or executable form.

The right to modify: This is the right to fix a problem or enhance a program. For most people, this means the right to hire someone to do the job, in much the same way one hires an auto mechanic to fix a car or truck or hire a carpenter to work in one's home. Modification is helpful. Application developers cannot think of all the ways others will use their software. Developers cannot foresee the new burdens that will be put on their code. They cannot anticipate all the local conditions, whether someone in China will use a program first written in Finland.

The right to redistribute: This means that you have the right to make copies of a program and redistribute it. You can charge for these copies, or give them away. Others may do the same. The redistributed code must include source code. Redistributed binary code only allows the user to run a computer. It traps him in dependence. This is not the way to achieve economic growth. Being able to create and distribute new code that fixes or extends older code also imposes a duty, which is to distribute the sources for the new code, under the same license as the older code. This means that people who use the new version of the older code retain the same rights and freedoms that they had when they used the older code. It means that if you fix my code, I have the right to use your fix. Most programs do have licenses. But some licenses do not impose this obligation to redistribute fixes or extensions.

Licenses, such as the famous BSD license, permit a person or company to take software that is itself free, and fix a bug or make an improvement, and then restrict who can use that fix or improvement. The United States government created the original BSD license. It was actually a way to subsidize partially monopolistic companies, since each received code that was paid for by the United States tax payer. The original Netscape Public License was also like this. The original source code can be viewed but if modifications or improvements were made to it, America On Line, the company that purchased Netscape, had the legal right to take the work and prevent the user from using any fixes to it or improvements to it that he made. That is they could legally prevent the user from using software with his own code in it! For success, a company must contribute to the community as well as take from it.

Redistribution brings competitive, free market collaboration

The right to redistribute, so long as it is defended and upheld, means that software is sold in a competitive, free market. Low price is one consequence of this. This helps consumers and consequently, the society. Primarily, these legal and economic rights lead to collaboration and sharing. This outcome is contrary to many people's expectations. Few expect that in a competitive, free market, every producer will become more collaborative and more sharing. Few realize that there will be no visible or felt competition among competing businessmen. The more competitive a market, the more cooperation is seen. This apparently counter-intuitive implication is both observed and inferred. Sharing occurs when people are not harmed by doing what they want to do. A competitive free market brings about cooperation. Visible competition indicates that the market is not fully free and competitive. If software is sold in a free market, competition among vendors will lead to a lower price. That is the price of software is determined primarily by legal considerations: by the degree of freedom that customers enjoy. If customers are forbidden to buy a product except at a high price, and that prohibition is successfully enforced, the product will be expensive. This is what occurs with most proprietary software nowadays. On the other hand, if software is sold in a free market, competition among vendors will lead to a lower price. This means that software itself, a necessary supporting part of a business or community project, will be both inexpensive and legal. Think of this from the point of view of a business or community supported group. The organization can use restricted-distribution, proprietary software, and either pay a lot of money it does not have, or break the law and steal it. On the other hand, free software is inexpensive and legal. It is more accessible. It is also customizable in ways that restricted software often is not. This is empowering.

The benefits of freedom (in terms of software)

What does freedom bring in terms of software? Programs are complex entities. They have thousands or millions of components. Because the components themselves are mathematical objects, that is to say, numbers and symbols, the components will not and cannot break, any more than the number 3 can break but the components can be combined wrongly, or inserted wrongly. Such bugs cause havoc. An advantage of free software is that lots of people have the opportunity to look at a piece of code. Somehow one of them will notice the problem and it will get fixed. In contrast, a proprietary company that sells updates will have a financial incentive to leave at least some bugs in its code. This ensures that customers will have an incentive to buy the upgrade. People purchase overpriced, buggy code, because they either do not

know about alternatives or they see what they are doing as less difficult than switching.

A notable feature of free software is that many applications run well on older, less capable machines. This frugality means that people can use older equipment. At the same time, manufacturers are building modern, low-end computers that do as much as the older ones, and are not too expensive. There is no need to acquire expensive hardware to run software. Free software brings with it frugal standards.

The benefits of freedom (to customers and businesses)

Freedom means that the customer has a choice among those who can provide software and associated services. Its not a 'take it or leave it' situation. The customer can choose among the vendors. While it is easier for a customer to leave, this also means that customers are not frightened of working with a small business that they like, but might collapse in five or ten years; the customer can move without trouble. The opposite situation can also occur: there can be the customer who decides to avoid a business because moving from it will be expensive, and the customer fears that the business will vanish in ten years. If its easy for customers to leave, then, employees know that they come to the business because the customers like the solutions the business sells. Employees like this, because it tells them they are doing a good job. Owners sometimes like this, too, since they too want to know they are living morally.

Freedom means that you, as a businessman, have the legal right to start a business. You are not hindered by overly expensive licenses. You are not forbidden. Likewise, as a customer, you may use the code. Freedom means that businesses are rewarded, with sales and profits, for satisfying customers legally, rather than rewarded by overcharging and hurting customers. Restricted software often means you are forbidden to start a business. Since free software is sold in a competitive market, its price is low. This means no one sells software as such. Instead, they sell services or they sell hardware as, for example, IBM does.

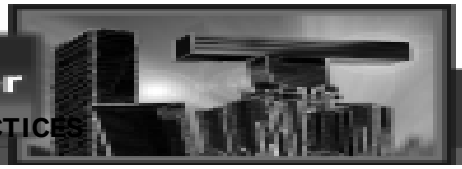
Success depends on satisfying customers. This makes both employees and customers happier. The alternative is policing, that is, making sure that software is not used or copied illegally. Generally speaking, the word 'policing' is not used. Instead, the trerm 'License Compliance' or some other such phrase is used. This practice is expensive and unpleasant. Software freedom create a world in which software does what the user wants. There is always the option that the user could write his own code (or hire someone to do it) if he cannot find an application that he needs. The user has the legal right, and with the source code, the practical right, to adapt other code to what he needs. This is often more efficient than writing from scratch. If a user does not want to spend the money and resources, then he can look around; often, it will be discovered that someone else has faced nearly the same problem, and has solution that can be used.

The benefit of freedom (ethical consequences)

Free software permits legal sharing. Yet, this is also an ethical issue. Do we want to encourage sharing or do we want to teach people to be selfish. Students in school like to give copies of programs to their friends. Often, this giving is illegal. The programs' distribution is restricted and the school children are supposed to insist that their friends, or the school, purchase additional copies. If you are a student, teacher, or administrator in a school, you can spend a great deal of time trying to enforce the law. On the other hand, you can teach your students to disobey the law. The solution is to adopt free software. Then students can be encouraged to give copies to each other: they can be encouraged to be both law abiding and to share with others. Students can also be encouraged to study the software that they have. Students can learn to program, to maintain systems, and they can learn to learn, which is very important in a changing world. Freedom brings the freedom to share. There is the legal right to help others. The legal right to collaborate. People who use binary-only software packages are forbidden to study them, learn from them, modify, or customize them. They gain no power from the software, except as far as the package itself solves a problem. Free software provides more than a solution; it provides the means for people to learn and become as good as or better than the programmers who wrote the software. It empowers people who previously were kept out of the circle.

Advantages to business

Free software imposes no legal barriers on the use of software. Free market means that software is inexpensive. Combined, these factors mean that free software reduces the 'barrier to entry' that often halts or prevents people from going into a new business. In addition to reducing barriers to entry, free software reduces costs of operation. From a managerial point of view a free software business is easier to run than a restricted distribution business. Free software requires less policing than proprietary, restricted code since the



product is sold in a competitive, free market where artificial restraints are not required. Also free software requires that a business focus be on selling solutions to customers rather than on policing them.

The idea that software should be sold has been described as a 'manufacturing delusion'. This is a business model. It is a decision to operate a business as if the software being distributed is similar to shoes or trucks. Software is not like a shoe or truck that is manufactured and then sold. As a practical matter, perhaps 3/4 of the costs for a typical software package come after the software is first released. This are costs of 'maintenance': the costs of adapting existing software to new hardware, the costs of debugging it, and the costs of extending the software to handle new tasks. A person who obtains a computer program does not want just the original, as with a pair of shoes or a truck.

The user wants the debugged versions, the extended versions. The 'manufacturing delusion' says to sell software at a high initial price, as if it were a truck or shoe, and then provide the fixes and improvements at little or no additional cost. This leads to disaster. For one, the owners of the software company see that fixes and improvements cost them money, rather than generate revenue.

Therefore, they cut back on fixes and improvements. Instead, they encourage their staff to focus on initial sales to generate revenue. However, existing customers then become upset and move to a competitor who offers a similar product that is better and since it is cheap to manufacture new copies of software, a competing company will reduce its prices to attract people to it. Customers will only stick to one company if they feel they have no choice: they will stay only if they see that the cost of changing is higher than the cost of staying. This means that a successful company must become a monopolist, and drive everyone else out of business, or at least, drive enough competitors out of business that the majority of its customers feel they have no choice of vendors. In addition, the company must make sure that no one else manufactures CDs with its software on it. Therefore, the successful monopolist will persuade its government to use its courts and police and foreign negotiators to prevent what it will call, dramatically, 'software piracy'. The 'manufacturing delusion' leads to catastrophe for all except the successful monopolist.

The Open Source/Free Software Business models

Because competition in a competitive market forces down the price of free software, no one should enter the software industry to sell software as such. Instead, a business should enter the industry to make money in other ways. In a free software industry, companies and people hardly sell software itself (manufacturers sell CDs with software on it, but the prices are reasonable). Instead, software companies and people sell services associated with software or hardware or other solutions. What type of services are these? Many companies help in using a computer, or, to take more specific examples, help in setting up a packet radio network, or help in creating and nurturing a warehouse database. Less directly, and increasingly, hardware companies that sell telephones or desalinization plants, add software to their products to make them more attractive to buyers. The most common software business is that of support: to introduce people to computers, teach them how to use computers, fix problems as they arise and to customize the software to local conditions. Free software distributions include programs for secure communication via email, publishing, browsing, budgeting etc. Training and support includes these as well as helping people to manipulate images, serve Web pages, or run an e-commerce site.

There are several business models proposed for new entrants into the software industry. One model is Paid-for Training. Private educational and training services provide quick profits for those who enter the business early. (Eventually, the ease of entry means that more enter the industry and profits decline). Another model is summarized by the phrase 'Give Away the Razor, Sell Razor Blades'. This describes the business model that the Gillette Company adopted a century ago for its razors. It did not quite give away the holder for its razor blades, but it sold them at a loss; and it made money by selling razor blades and it still does. People have paid the Gillette Company far more for the razor blades than for their razors. Ron's DataCom and other free software companies, like Red Hat, also use the software as a 'Market Positioner': the software brings people to them to purchase their other services. A third business model has Free software companies sell a brand, like EasyLinux or Red Hat. The companies get paid for providing a trusted product. This depends on having a known and good reputation. Companies can do this in two ways: one, quite obviously, is to sell a software distribution. Customers know the company selected the software and did a good job, so the customer does not have to do the work. A second, more subtle way to sell a brand is to sell certification: to guarantee to others that some other product is good. Besides selling services, or selling a brand, or selling the value inherent in a complete system, businesses can sell other kinds of products. A fourth business model is 'Selling an adjunct'. This kind of product is that which goes with or explains a program. For example, O'Reilly

sells computer books. Similarly, a computer manufacturer can put together hardware, or recondition old machinery, and load it with inexpensive, customized, free software. (Again, early entrants can make large profits, before the industry matures). A fifth business model is 'Widget Frosting'. This is similar to 'Selling an Adjunct', except that the product sold is more important than the software. In English, a widget is an unspecified, manufactured object. Frosting is what you put on a cake, to make it tastier. 'Widget Frosting' is the process of making a manufactured object more desirable to customers. IBM, a large corporation, found that some of its customers refused to buy bigger and more expensive computers from IBM, even though they needed the larger capacity. The customers were afraid that their existing software would not run on the bigger machines. Hence IBM has adopted GNU/Linux to its whole range of hardware from its smallest laptop to its largest mainframe. As a result, an IBM salesman can say 'look, GNU/Linux runs on the machine you are using now; and it runs on this bigger machine. Your software will run, too. So you can buy the bigger machine safely.' IBM uses the software to sell its hardware. It is important to note here that most software is not written for sale, and never had been. Many people do not realize this. Instead, most software is written for use in other products, like airplanes or ships, or in business or database systems. On its own, none of this software has what might be called a 'sale value'; it has only a 'use value'. In the US for example, less than 10% of all software is written to be sold [11], [12], [13], [14], [15]. However, the software that most people think about is sold under the 'manufacturing delusion'. It is visible. People who see a PC often think of the software on it. People who see a washer or truck seldom think of the software in it.

Companies that manufacture trucks or washing machines or electric generating plants often use the 'Widget Frosting' business model. They create software that runs inside their products (embedded software) and thereby make their products better than they would be otherwise. There are two reasons such companies adopt free software. First, free software provides the company with an existing, complex system that works. The companies need to do less work of their own. It costs them less. Second, the free software leads to better products, so customers like them more. So the companies sell more. Of course, other companies can use the same software: you need to give people a reason to buy from you. Here is where virtue becomes profitable; people will buy from you if your hardware is better, or your service is better, or if they like you for some other reason. And, of course, if you are not well known, people will be more likely to risk buying from you if they know they can hire someone else to work on your product. You reduce your customers' risk by providing them with free software. It is the paradoxical rule: if it is easy for your customer to leave you, your customer is more likely to stay.

Creating Value from Free Software in a developing country

Economic growth and development is often an elusive goal. Thus, when a new source of economic development is offered, it attracts attention. In recent years, several policy advisors have offered OSS/FS as a source of economic growth and development. In developed nations, especially in the United States (U.S.), software has made a significant impact on the economy. If the U.S. experience could be translated to other countries, even on a smaller scale, the potential impact could be large. Hence, the interest in a low-cost means of developing a local software industry is easily understandable. As a percentage of total GDP, the software industry in the U. S. is actually relatively small. Nonetheless, as a percentage of total exports the effect of the software industry on the economy is much higher. The software industry ran a trade surplus of $13 billion in 1997; without software's contribution, the U.S. trade deficit would have been 36 percent higher [16]. However, a digital divide continues to exist between developed and developing countries. Considering for example China's software industry, which is still relatively small, it constitutes roughly 2% of the Worlds software industry, whereas the U.S. market share is 40% and Europe's market share is 31%. In the U. S. 97% of software is provided by local companies whereas in China local companies provide only one third. In 2002, China's total software industry revenue is about $13.3 billion (USD), which is insignificant when compared with developed countries when considering China's 1.3 billion people, which constitutes more than one-fifth of the world's population [17].

It is common to have people refer to the Indian software industry and many people think the Indian software industry is successful. The Indian software industry is mainly outsourced work from other countries [17]. However, India, which is largely a Microsoft environment, is also adopting OSS/FS because with limited funds to support ICT development, they do not want to become vulnerable to sudden, uncontrollable changes in pricing by being tied to one vendor. In addition, there is the need for Indian scientist and engineers to have the technical knowledge to develop and maintain their own systems without being dependent on any vendor [18]. China and Peru have also raised security and transparency issues as arguments for open source deployment, as they can

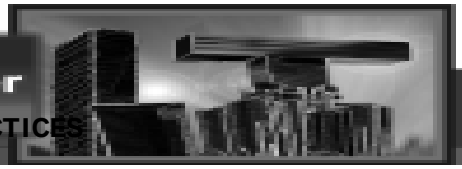
exert control over the use of open source encryption methods. In the case of countries saddled with large foreign debt payments, such as Argentina, policies that favor OSS/FS have been adopted because of expected cost savings.

Some key issues that have been raised that are vital for establishing the worth of Open source in the context of development. These are: can developing countries create value through OSS? What are the barriers to OSS/FS? Moreover, what practical approaches can be encouraged? There is a need to address these issues if OSS/FS is to develop and have the potential to bring benefits to the economy and society. The main constraint incidentally is that which provides most open source software its legal backing. The famous GNU GPL licence. If a program is distributed under the GPL, all source code must be made available, free. The GPL also stipulates that any user can modify and distribute the program, either in original or modified form. Any redistribution, though, (whether of the original or modified program) must also come under the GPL. This condition has earned the GPL the label of "viral" because it typically means that once code is licensed under the GPL, any other program that incorporates that code falls under the GPL as well. The GPL provisions were intentionally aimed at preventing open-source code from being incorporated into proprietary code. One result of the source code distribution requirements is that programmers can charge no more for programs than the cost of reproduction (which are typically quite small). If a programmer tried to charge license fees substantially above the reproduction costs for GPL software, anyone else could acquire the source code and redistribute it on their own, driving the price back down to reproduction costs. With license fees thus foreclosed, the only profit opportunities remaining are for additional services, such as software support or training, or for complementary proprietary programs that run with or on the open-source program. The licensing provisions clearly have implications for firms hoping to earn a sustainable return on software production.

While it has been shown that that open source enables a country to develop its local software industry without having to tackle thorny intellectual property rights issues. However, can the fact that open-source software can avoid some intellectual property issues answers the question whether it has the ability to promote economic growth? The case for open-source software as a growth and development tool is weak. Often, the arguments muddle reasons for using the software with reasons the software might promote economic growth. The low initial cost of open-source software, the freedom it affords from Western-based companies and the opportunities it can provide for local programmers are all valid points however, none of them show the ability of open-source to spur economic growth or even to its ability to establish a viable local software industry. Regardless of the specifics, the underlying economics imply that *pure open-source software production* cannot generate sustainable profits, which is an important point to bear in mind when considering the arguments for governments to use open-source as a development tool [16].

IBM has been cited as the epitome of a profitable firm with open-source offerings. In 2001, IBM spent $1 billion (USD) backing Linux [3]. In 2002, it announced that it had recouped this investment in full. Certainly, IBM's highly visible support of open-source software has been profitable for the company. IBM is not, however, primarily a software company. It is a services and hardware company that has successfully deployed Linux as a means to sell its services and hardware as well as its proprietary software. Thus, IBM's experience does not provide developing nations with a road map to large financial rewards via open-source software. To even attempt this route to economic growth, countries would first need to foster a high tech hardware industry, along with a services and proprietary software industry. The most prominent example in the developing world of a newly emergent software industry is India, and this was without any open-source contributions. The rapid growth of India's software exports, which comprise 70% of its software industry, is due to its comparative advantage in labor. India has a large reserve of well-qualified, English speaking engineers and technicians that it has parlayed into outsourced proprietary software production for mostly Western clients. These particular circumstances raise the question of whether India's experience can be replicated among other developing countries. Only a few developing countries, such as Russia and China, have a larger reserve of engineers than India. These countries have other disadvantages, though, such as a lack of international language skills [16].

While the interest of developing nations in open source is understandable, given its low investment costs and the overall appeal of software as a source of revenue, the *pure open source model* on its own does not appear to provide a solid foundation for profitable business operations that can meaningfully contribute to a developing nations' economic growth. The pure open-source model is not capable of supporting proprietary software firms. While the service-support model can provide sustainable profits, as the U.S. experience has demonstrated this model can only support a handful of firms at best. Proprietary software applications designed to run on open-source software (operating system)



appear to be the most viable profit-making option (like IBM). Since hardware and software production can be separated, developing nations can complement the progress of developed nations without necessarily entering into direct competition with them.

The Nigerian Software community needs a legal and institutional framework to protect and preserve the rights of the programmers. Without law, there are no practical sanctions and the agents of the law must be reliable, quick, and honest. One legal tool is a specially drafted copyright license, to protect and preserve free software (just like some of the hybrid licensing schemes that have been proposed and used to license OSS/FS software in some developed countries). This is not a technical or business issue: what makes software free rather than imprisoned is the legal and institutional framework in which people work.

Conclusion

Today and in the future, Information Technology promises to change the landscape of the Nigerian Business, Social and Economic communities. Anyone who own or uses a computer system deserves to be able to cooperate openly and freely with other people who use software. The programmer and users deserve to be able to learn how the software works, and to teach their students or protégés with it. Not being a programmer is not a constraint, you deserve to be able to hire your favorite programmer to fix your program when it breaks down.

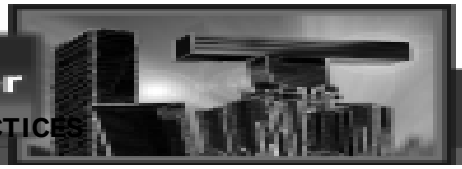
The achievement of economic growth for any nation depend on the opportunities to do business which in turn depend on the legal and practical freedom to: copy, study, modify, and redistribute software under a free license. The key is freedom. Freedom leads to collaboration, lower prices, reliability, efficiency, security, fewer barriers to entry, and fewer barriers to use, more opportunities in business. Freedom leads to a better society.

REFERENCES

1. Chassell, J. R. “Software Freedom: Rights, Duty, Metaphor, and Making a Living” Free Software Foundation - GNU Project. (http://www.zope.org/) downloaded October, 2004
2. Joanne Richardson (2001). Free Software & GPL Society — Interview with Stefan Merten, Oekonux (DE). http://subsol.c3.hu/subsol_2/contributors0/mertentext.html. Downloaded November, 2004.
3. Chassell, J. R. “Free Software Foundation - GNU Project.”. http://www.freebsd.org/) (http://www.freebsd.org/http://www.gimp.org/). Downloaded October, 2004
4. David A. Wheeler, 2003, “Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers!” (http://www.dwheeler.com/oss_fs_why.html). Downloaded January 2004.
5. Steve M., 2004, “Open Source Definition.” Copyright © 2004 The Open Source Initiative, (http://www.opensource.org/docs/definition.php), Downloaded January 2004.
6. “Philosophy of the GNU Project”, 2003, Copyright (C) 1999, 2000, 2001, 2002 Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111, USA , (http://www.cs.waikato.ac.nz/~daven/docs/oss-wp.html), Downloaded January 2004
7. John Carroll, 2003, “In defense of proprietary software”, ZDNet, (http://zdnet.com./2100-1104_2-5125160.html), Downloaded January 2004
8. Philippe Aigrain (2002) A framework for understanding the impact of GPL copylefting vs. non copylefting licenses.
9. Andy Tai (2001). Free Software: Hackers Comeback. http://.Downloaded October, 2004.
10. Lerner, J. and Tirole, J. (2000). The Simple Economics of Open Source. Working Paper 7600. http://www.nber.org/papers/w7600. Downloaded October, 2004.
11. Richard Stallman (2001). “Why Software Should Not Have Owners”. http://www.rons.net.cn. Downloaded October, 2004.
12. Richard Stallman (2001). “ Copyleft: Pragmatic Idealism”. http://www.rons.net.cn. Downloaded October, 2004.
13. Nathan Newman, 1999, “The Origins and Future of Open Source Software”, A NetAction White Paper, (http://www.netaction.org/index.html), downloaded January 2004.
14. Eric Lease Morgan, E.L. (2003). “Open Source Software in Libraries: A Workshop”. http://infomotions.com/musings/ossnlibraries-workshop/

http://infomotions.com/musings/ossnlibraries-workshop/. Downloaded January 2004.

15. *Hecker, F. (2000).*Setting Up Shop: The Business of Open-Source Software http://www.hecker.org/writings/setting-up-shop.html.
16. Bibek Debroy, Julian Morris,2004, “Open to Development: Open-Source Software and Economic Development”, www.policynetwork.net/pdfs/open-to-development-2004.pdf, [Downloaded Jan 2005]
17. Li Wuqiang,2003, “Open Source Software and China’s Software development” Strategies for Building Software Industries in Developing Countries, www.iipi.org/activities/forums/softwareconference/Li%20Paper.pdf, [Downloaded Jan 2005]
18. Dinesh C. Sharma, 2004, “Indian president calls for open source in defense”, CNETAsia, http://asia.cnet.com/news/software/0%2C39037051%2C39186139%2C00.htm, [Downloaded Jan 2005]



Users and Developers Expectations in Software Products

Zsolt Lipcsey, Edem Efiong Williams, Eyo Okon Ukem, Rufus Chika Okoro

ABSTRACT

Software development is business oriented and one expects some measure of profit margins on the side of the developers and assurances on the side of the users. Software products are important in Information Technology (IT), in that the functionality of a computer system depends largely on the quality of the software installed. Like any other business, the products of software development must be competitive in the market in order to achieve the above stated goal of remaining in the business. This implies that software quality must be evolving beyond static assessment to include behavioural attributes, such as availability and maintainability. Developers and users therefore would expect that the software meets certain standards and provides satisfaction. This paper gives a survey of these behavioural attributes, the various factors that can lead to both the developer and users deriving a reasonable measure of satisfaction from the software product. The paper also tries to analyse a group of quality parameters and model their relationships.

Key Words: software, reliability, maintainability, portability, usability, cost-effectiveness, compatibility, efficiency.

INTRODUCTION

Software products are important in Information Technology (IT) in that the functionality of a computer system depends largely on the quality of the software installed. It is therefore very important for a user to be able to rely on the software used in the computers. In this era of monetary acquisitions by many people in the technology, there is bound to be varying levels of quality in the production of computer software. This varying quality of software invariably affects the performance of the computer. The user therefore would not be happy to have a computer system perform below expectations. It is therefore necessary for the users that the software used in computer systems be assured to have certain characteristics.

When these characteristics are to be formulated, it has to be taken into consideration that there are various types of users; there are the eager users and the reluctant users, the expert users and the novice users, the major users and the minor users, the professional users and the amateur users. All categories of users have their expectations, which are invariably included in the assurance of quality of the products they use.

The performance of a functional computer system depends on two broad categories of products – hardware and software. While the assurance of the quality of the hardware is the sole responsibility of the equipment manufacturers, the assurance of the quality of software is a responsibility of a large variety of software developers.

Since the user requirements depend on a large variety of users of different categories, and also many developers with varying views produce the software, it is the professional's duty to fill with contents the concept of assurance of quality of software for the client. The assurance of computer software covers the qualities of upgrade-ability, maintainability, compatibility and portability just to mention a few important ones. Any software that runs short of these can be regarded as not reliable and its functionality will be in doubt as related to efficiency. Most software in the market today are pirated copies, which sometimes does not contain the vital information about the software, for example the program keys (CD Keys or program numbers). It is obvious that some software cannot be effectively installed without this program keys and if for any reason one manages to install this pirated copy of the program, the efficiency of the software will certainly be low. The paper therefore seeks to address the evolving characteristics or attributes of software products, address needed user characteristics, and also address the developer's expectations.

USER EXPECTATIONS

Software acquisition and usage usually involves expenses. Users therefore would expect that the software meets certain standards and provides

satisfaction. The satisfaction can be considered broadly under the following sub-headings.

Reliability (Rb): Reliability of a software product can be considered to be the extent to which it can perform expected functions with required precision over a given period of time. It would be very disappointing for a user to purchase software only to discover that the software is not performing to expectation. Every software system or product exists to provide value to its users (Hooker, 1996), therefore the developer should have this in mind as a legitimate expectation of the user.

Maintainability (Ma): In software usage there is often the need for changes to be made to accommodate new functions in a particular environment. The process of changing a system after it has been delivered and it is in use is called software maintenance. The changes may involve simple changes to correct coding errors, more extensive changes to correct design errors or significant enhancement to correct specification errors or accommodate new requirements. Sommerville (1996) considers maintenance in this context to mean software evolution.

Portability (Po): Portability is defined as the ease of transporting a given set of software to a new hardware or new operating system environment. Software forms the basic item for system functionality. User expectation therefore is that the software purchased could be installed on any alternative system as the need may arise with due regard to copyright protection.

Usability (Us): The use of software should not need sophisticated knowledge to operate. In this context a minimal idea of the software should be enough. Software design is not a haphazard process. There are many factors to consider in any design effort, and all design should be kept simple, so as to facilitate having more easily understood, easily operated, and easily maintained systems (Hooker, 1996). However, there should be an easy to understand users' manual for referencing.

Reusability (Ru): Reusability involves the use of most modules from an existing software for the development of a new one. For a developer therefore, to effectively develop continuously he will need to pick up some modules from already existing software in form of upgrading. Otherwise it will be wasting efforts to duplicate modules in software development. Reuse saves time and effort, but achieving a high level of reuse is the hardest goal to accomplish in developing a software system. The reuse of code and design has been proclaimed as a major benefit of using object-oriented technologies (Hooker, 1996). In expecting reuse, the developer should bear in mind that planning ahead for reuse reduces the cost and increases the value of both the reusable component and the systems into which they are incorporated.

Cost-effectiveness (Cs): For any user who spends money for software would certainly want to get the result of the expenses. It is actually this desire that makes him/her achieve his goals for which purpose he/she procured the software. There is no gain saying that any software that does not justify the expenses on it is useless. Cost effectiveness has to do with the quality of an item measures by its ability to satisfy a solution in a given period.

Compatibility (Co): Compatibility has to do with the ability of software version to be substituted for any other lower version, achieving the same goal. In this effect the user will always expect to have this upwards compatibility. Otherwise he/she will tend to believe that the software is not meeting his/her desired goal. Since the software technology is dynamic there is the tendency that newer/higher versions of software will always be available in the market and if a user is unable to change to this newer/higher version without affecting his data/information then the software is unsatisfactory. For the user this will be a wasted effort.

Efficiency (Ef): Efficiency is the extent to which software uses minimum hardware resources to perform its functions. Although , many software require a particular hardware standards for operation it is therefore important that a software developer takes this into consideration during the process of developing a software with view to producing that which will make use of the minimum hardware resources in a given computer system for operations.

Users want to buy quality software. Quality (Q) can be used in place of satisfaction. Software quality is no more considered in terms of static assessment of the code structure. Jeffrey and William (2004) redefine software quality to include also the non-functional or behavioural attributes of reliability and maintainability. From Jeffrey and William's redefinition of quality, the developer must consider some set of key behavioural attributes such as reliability (Rb), performance (P), fault-tolerance (F), safety (Sa), security (Se), availability (A), testability (T), and maintainability (Ma). Software quality (Q) is therefore a function of these combined attributes plus an error term, e, representing the quality aspects which these attributes can define.

$$Q = f(Rb, P, F, Sa, Se, A, T, Ma) + e.$$

As rightly observed by Jeffrey and William, the quality equation is impracticable in its current form, but it serves as a starting point to use for examining the elements, which software developers should consider globally and

in Nigeria in particular.

The various software quality measures contrast each other in the sense that making any of them absolute requirement and producing a software perfect in that sense may lead to a product which is either not operational, extremely costly or brings in grossly unsatisfactory attributes. For example, software, which is over-sophisticated, may cost unreasonably high, a perfectly portable software may have compromises which reduces performance etc. Another example is that as security increases, performance decreases, because increased security consistently reduces performance. Also, as fault tolerance increases testability decreases because if software can more easily reveal hidden defects at test time, failures are more likely to propagate when the software is operational. Hence we are facing a typical game theoretical situation in the case of software quality measurement: The various requirements cannot be satisfied to the extreme; hence we want the best acceptable compromise. Hence if we want to model software quality, we have to set up a game theoretic model for it. This means that to formulate the various quality functions in terms of observable random variables and after establishing their interactions, we may be able to determine how far a product is close to what we may call the best acceptable compromise. Such compromise would be a reflection of the application environment.

DEVELOPERS EXPECTATIONS

Software developers also have expectations. Losing user loyalty is a problem that companies take seriously, especially in a dynamic market where customer allegiance lasts only until the competition produces a more user friendly version. For example, Merril and Feldman (2004) suggest that users flocked to Google almost overnight because the company understands the logic of how people want to search the web. Therefore every producer wants to satisfy the user, stay in the and make his profit.

Normally, the software developer will be concerned with safety, mission criticality of the software, size and complexity of the product, size and operational complexity of the development staff.

Safety here means reducing risk factors, which include among others time schedule that can be kept, specifications that assure a working solution.

The mission criticality refers to the importance of the product – how critical a problem is that is to be solved.

Budgets and schedules are closely interrelated. Tight budgets and tight schedules normally go together.

A clear vision is essential to the success of a software product, so there is need to maintain vision. Without conceptual integrity, a system threatens to become a patchwork of incompatible designs held together by the wrong kind of screws (Hooker, 1996).

The developer expects the product to have long life, because a system with a long lifetime has more value. However, in today's computing environment, software lifetimes are typically measured in months instead of years, due to the fact that specifications change rapidly and hardware platforms become obsolete when just a few months old as a result of fast moving technology. In order for software products to enjoy long lifetimes, therefore, they have to be designed in a way that makes them adaptable to these and other changes. The design should be future-looking (Hooker, 1996).

SOCIAL CONSIDERATIONS: THE NIGERIAN SITUATION

Admittedly, Nigeria is relatively new to IT applications. As such, the generality of Nigerians are not well placed to reap the benefits of the IT revolution. Some of the reasons for this situation are poverty, illiteracy, ignorance, and inertia. Further expectations of software users in Nigeria would be:

Low Cost: The cost of software should deliberately be made to be reasonably low, so that more Nigerians would be able to afford the products, thereby permitting more Nigerians to computerize.

Simplicity: Because of the level of literacy in the country, which is generally low, software products should be kept simple. This way more Nigerians would be attracted to software applications and would not be intimidated by them.

Awareness: Software developers, in conjunction with such bodies as the Nigeria Computer Society and the Software Practitioners Association of Nigeria, should create proper awareness in the software market place.

Education: Software users in Nigeria, perhaps more than in some other places, need to be much more educated on the capabilities of the products in existence, and generally on the inherent advantages of computerization or automation.

Adaptation: Software designed for use in Nigeria should be adapted to the locality, by as much as possible reflecting the good customs of the land. This way Nigerians would feel more at home with such software.

From the above survey of the Nigerian peculiarities, we can say that a developer who wants to stay in the Nigerian market should give attention to these factors.

CONCLUSION

Every software product is developed with a definite aim and for a definite purpose. In the process of the development, the developer has to consciously take measures to ensure that the product meets set requirements. The developer expects that others will ultimately use what he develops. Hardly is a software product constructed and used in a vacuum. In some way or other, someone else will use, maintain, document, or otherwise depend on being able to understand the product. Thus the audience for any product of software development is potentially large, and the developer bears this in mind, and expects to add value to the system by making the jobs of this user audience easier. He has to ensure that whatever standards are in existence are adhered to in the process of development.

REFERENCES

Carla Merill and Diane Feldman (2004) Rethinking the Path of Usability: How to Design What Users Really Want, IT Professional, IEEE.

Hooker, David. 1996. Seven Principles of Software Development.

Jeffrey Voas and Williams W. Agresti (2004): Software Quality from a Behavioual Perspective, IT Professional, IEEE.

Jorgensen, Magne and Sjoberg, Dag I. K. (2003) Impact of Customer Expectations on Software Development Effort Estimates.

Software Assurance guidebook NASA GB-A201

Sommerville, Ian (1996) Software Engineering. Addison Wesley Publishing Company.

Risk Management Model: Techniques and Application to Information Technology Management

Peter O. Olayiwola

A B S T R A C T

Risk management provides a structured approach to decision analysis. It is about minimizing the probability of loss and maximizing the chance of success of your decisions. Operational risk management is about operational excellence and it follows the plan-do-assess-adjust process. These four steps will be adequately covered in the paper with particular application to Information Technology. The process begins with identifying the sources of risk (or hazards) for the operation. This can involve all sources of potential loss to the business, including business interruption, damage to hardware/software, the facilities, the corporate image, property and other assets or things of value. Most methods of hazard identification are qualitative in nature. The end result of the risk identification process is an inventory of the organisation's sources of operational risk. The next step is to quantify the risk associated with each hazard. This provides a basis for setting priorities in terms of which risks are acceptable, which ones need to be addressed immediately and which ones are a lesser priority. Risk mapping is an effective tool to estimate the risk that various hazards pose. Risks are plotted on a matrix (Risk Matrix) with consequence likelihood (or expected frequency) along one axis and consequence severity along the other. Participants will be taken through an exercise on how to develop a Risk Matrix. After you have completed a risk profile, you will never look at the information technology operations of your organisation in quite the same way again. You will be keenly aware of how risks and opportunities can arise, and what you need to do to effectively and efficiently manage your operational risks so that your organisation can continue to grow and thrive.

1.0 INTRODUCTION

Risk is typically defined in terms of the potential downside of a decision or action. *It is commonly described as probability of a loss or the combination of the probability of an event and its consequence.* However, the essence of modern risk management is as much about taking advantage of opportunities to prosper as it is about avoiding potential losses. With every action we take or don't take, there is risk - positive and negative. On the upside, the opportunity is there for benefit; on the downside, there is the threat of failure. Risk management provides a structured approach to decision analysis. *It is about minimizing the probability of loss and maximizing the chance of success of your decisions.* If no decision needs to be made, then there is probably no risk.

There are *two dimensions* of risk analysis: *quantitative* - how big are the risks associated with a decision, usually expressed as the probability and magnitude of potential benefits and consequences - and *qualitative* - what the decision and the range of possible outcomes mean for the organisation and its stakeholders, given the values, goals, needs, issues and concerns of each party.

Traditional risk management disciplines, such as insurance and finance, tend to focus on the quantitative elements of risk analysis. However, ignoring the qualitative or value-laden side of risk can be disastrous. One example is the failure of a political party in a geo-political zone of Nigeria to return most of its governors in the 2003 general election by its failure to present its own presidential candidate.

The field is of growing interest to public and private sector managers. One driver is the demands by the public and the shareholders for well-run operations that achieve their objectives and avoid losses. Twenty years ago, the risk manager was the organization's insurance buyer. Today, risk management professionals deal with many issues that fall outside the scope of an insurance policy, such as threats to reputation, demands for due diligence and communications with stakeholders about the risks and benefits associated with the activi-

ties of an organisation.

A growing number of organisations are embracing enterprise risk management, which involves ensuring all risks across the organization are managed in a consistent and disciplined way. It is an *holistic approach* that considers *strategic* (what products/services should we offer to achieve our goals?), *financia*l (how can we protect the organisation against currency/monetary risks?) and *operational concerns* (what loss prevention programmes and contingency plans have we put in place?).

2.0 OPERATIONAL RISK MANAGEMENT

Operational risk management is about operational excellence and it follows the *plan-do-assess-adjust process* that is used for quality management. There are four steps to the continuous improvement process that is applied to operational risk management:

- **Planning** involves analysis to identify risk exposures and what they mean to the organisation's bottom line. It also involves assessing the current level of risk controls and deciding what actions are needed to address any gaps in the risk management program that could prevent the organization from achieving its business goals (ensuring the right things are done).
- **Doing** involves the implementation of risk controls and other risk management activities.
- **Assessing** involves monitoring and auditing to ensure that risk controls are carried out as planned (ensuring things are done right).
- **Adjusting** It is critical that senior management conduct periodic reviews to ensure that its risk management program continues to meet the organisation's needs and that necessary changes are made.

The planning stage typically includes the following steps:

- **Hazard identification**. Various hazards or sources of risk are identified. The end result is an inventory of risk sources.
- **Risk assessment**. The organisation assesses the likelihood and potential severity of consequences for identified risks. It also determines its risk tolerance levels that reflect its appetite for risk.
- **Gap analysis.** When significant risk is involved, the effectiveness of up-to-date risk management controls for specific risks is assessed. The strengths and the weaknesses in the organisation's overall continuous improvement cycle for operational risk management are identified.
- **Risk control options evaluation.** Beginning with the higher risks that are assessed as poorly managed in the gap analysis, risk control options are generated and evaluated in terms of potential risk reduction, resources required and other costs and business benefits.

The process begins with identifying the sources of risk (or hazards) for the operation. This can involve all *sources of potential loss* to the business, including business interruption, damage to health, the environment, the corporate image, property and other assets or things of value. Most methods of hazard identification are qualitative in nature and benefit from a team approach that brings in representatives who have knowledge of all aspects of the organisation's operations. Checklists are an effective way to ensure familiar risks are identified. A series of *"what if" questions* can help uncover latent risks. Regardless of the method, the risk analysis team should start by reviewing the operation through a site tour that is combined with a review of documentation, including the organisation's risk management policy, operations procedures and any other available risk information (such as summaries of past insurance claims, health and safety incidents, downtime or employee satisfaction surveys). The end result of the risk identification process is an inventory of the organisation's sources of operational risk.

The *next step* is to *quantify the risk associated with each hazard.* This provides a basis for setting priorities in terms of which risks are acceptable, which ones need to be addressed immediately and which ones are a lesser priority. Risk mapping is an effective tool to estimate the risk that various hazards pose. (See Figure 1: "Risk Matrix" below) *Risks are plotted on a matrix with consequence likelihood (or expected frequency) along one axis and consequence severity along the other.* The risk map example shows three categories to rank expected frequency: *Unlikely, Occasional and Often*. The consequence axis can reflect a wide range of potential consequences but for the example shown, the consequence severities are: *Low, Medium, and High.*

The organisation's risk tolerance is codified when it documents what risk level it assigns to each cell in the risk map Just as each organization has a unique risk profile, it will also have a unique *risk tolerance. A organisation's appetite for risk is linked to corporate val-*

ues, objectives and culture. It is critical that the leadership team determines criteria for risk tolerance and provides the commitment, support and resources to managers to develop and implement risk management activities.

HIGH	6	8	9
MEDIUM	3	5	7
LOW	1	2	4
	UNLIKELY	OCCASIONALLY	OFTEN

FIGURE 1: RISK MATRIX

3.0 RISK MANAGEMENT MODEL: APPLICATION TO PROJECT MANAGEMENT

Risk is a major factor to be considered during the management of a project. Project management must control and contain risks if a project is to stand a chance of being successful. This section covers the main aspects of the management of risk as they apply to project management.

3.1 WHAT IS RISK AS APPLIED TO PROJECT MANAGEMENT?

Risk as applied to project management can be defined as the chance of exposure to the adverse consequences of future events. Projects are set up to bring about change, and hence project work is less predictable than is typically the case with non-project work. The project is unique and usually its objectives have to be achieved within certain constraints. In addition, projects can be large and complex, and can deal with novel or unusual factors. There are a number of 'standard' risks that are most likely to affect all projects. Some examples of these appear below. There will be other risks that are project-specific, in that they exist because of a particular project and its circumstances. Management of risk is an essential part of project management. There are a number of techniques available that can be used to achieve this.

3.2 TYPES OF RISK

Broadly, there are two types of risk - *Business risk and Project risk.*

3.2.1 Business Risk

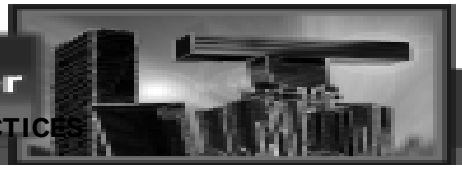
This covers the threats associated with a project not delivering products that can achieve the expected benefits. It is the responsibility of the Project Steering Committee to manage business risks. It includes such areas as:

- the validity and viability of the project
- whether the project continues to support the corporate business strategy, including such elements as: strategic direction, commercial issues, and market change
- the consequences to the corporate body of failure or limited success
- the stability of the business areas involved
- programme requirements
- legislative changes
- political factors, including public opinion
- environmental issues
- the impact on the client of the results of the project
- the risks of the end result meeting the stated requirements but not fulfilling expectations.

3.2.2 Project Risk

This is the collection of threats to the management of the project and hence to the achievement of the project's end results within cost and time. These risks may be managed on a day-to-day basis by the Project Steering Committee, Project Manager or Team Manager or Team Leads. Risks will be many and varied, but would include the following broad categories:

- Supplier issues, covering those risks caused by being dependent on a third party, including:
  - failure of the third party
  - failure by the third party to deliver satisfactorily
  - contractual issues
  - a mismatch between the nature of the task and the procurement process



- Organisational factors such as:
  - additional staff responsibilities alongside project work
  - the project culture, or lack of it, within the client organisation
  - personnel and training issues
  - skill shortages
  - potential security implications
  - culture clashes between client and supplier
- Specialist issues: there will be a wide variety of issues here because each project has its own particular specialist elements, which bring with them their own risk elements. However, there are some general issues that will apply to many project types, such as:
  - how well requirements can be specified
  - to what extent the requirements can be met using currently available and understood facilities and approaches
  - the extent to which a project involves innovative, difficult or complex processes and/or equipment
  - the challenges and problems regarding quality testing
  - the risks that the specified requirements will not be achievable in full, or that not all requirements will be correctly specified.

It must be stressed that the above lists are given purely to illustrate the areas of risk that need to be considered as part of project management. Each project must be considered in its own right. Once identified, risks are not kept separate. Both business and project (and other relevant) risks are all entered in the one Risk Log, which is always reviewed in its entirety.

3.3 MANAGING RISK

The management of risk is one of the most important parts of the jobs done by the Project Steering Committee (a committee set up by the client to oversee the project) and the Project Manager (a professional Consultant or Consulting firm). The Project Manager is responsible for ensuring that risks are identified, recorded and regularly reviewed. In this regard, the Project Steering Committee has four main responsibilities:

- notifying the Project Manager of any external risk exposure to the project
- making decisions on the Project Manager's recommended reactions to risk.
- striking a balance between level of risk arid the potential benefits that the project may achieve
- notifying programme management of any risks that affect the project's ability to meet programme constraints.

It is the responsibility of the Project Manager to modify plans to include agreed actions to avoid or reduce the impact of risks. An 'owner' should be identified for each risk, who should be the person best situated to keep an eye on it. The Project Manager will normally suggest the 'owner' and the Project Steering Committee should make the decision. Project Steering Committee members may be appointed 'owners' of risks, particularly risks from sources external to the project.

In order to contain the risks during the project, they must be managed in a disciplined manner. This discipline consists of:

- risk analysis, which involves the identification and definition of risks, plus the evaluation of impact and consequent action
- risk management, which covers the activities involved in the planning, monitoring and controlling of actions that will address the threats and problems identified, so as to improve the likelihood of the project achieving its stated objectives.

The risk analysis and risk management phases must be treated separately, to ensure that decisions are made objectively and based on all the relevant information.

Risk analysis and risk management are interrelated and undertaken iteratively. The formal recording of information is an important element in risk analysis and risk management. The documentation provides the foundation that supports the overall management of risk.

Risk analysis requires input from the management of the organisation. The organisation's management, in turn, is kept informed by the analysis in a highly iterative manner. Communication is particularly important between the project and programme levels within the organisation.

Where the project is part of a programme, the management of risk procedures used by the project must be consistent and compatible with those of the programme unless there are valid reasons not to do so. Where a risk is uncovered in the programme, any affected projects should be involved in the analysis of that risk. Similarly, project risk evaluation should include staff from the programme. Project risks that threaten programme milestones or objectives must be escalated to programme management.

CONCLUSION

After you have completed a risk profile, you will never look at the operations of your organisation or a project management assignment in quite the same way again. You will be keenly aware of how risks and opportunities can arise, and what you need to do to effectively and efficiently manage your operational, business and project risks so that your organisation and clients can continue to grow and thrive.

Business Process Reengineering (BPR)

Opara Pascal

ABSTRACT

A business process is a set of activities that transform a set of inputs into a set of outputs for another person or process using people and tools. Improving business processes is paramount for businesses to stay competitive in today's marketplace. Many companies began business process improvement with a continuous improvement model. The model attempts to understand and measure the current process, and make performance improvements accordingly. This method is effective to obtain gradual and incremental improvement but not enough because new technologies and globalization have raised the competitive bar. In today's marketplace, major changes are required to just stay even. As a result, companies have sought out methods for faster business process improvement. One approach for rapid change and dramatic improvement that has emerged is Business Process Reengineering (BPR). Business process re-engineering is the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical, contemporary measures of performance, such as cost, quality, service and speed. Information Technology plays a central role in Business Process Reengineering through process automation, simplification and improvement. The Microsoft Solution Framework, Alchemy Approach, is one of the best in class software engineering process and practices for early realization of BPR project benefits.

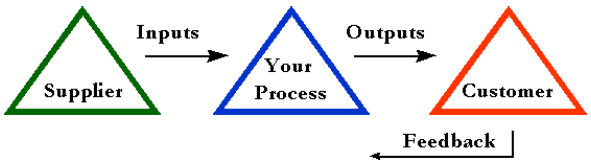
Introduction and Concepts

If you have ever waited in line at a super market store, you can appreciate the need for process improvement. In this case, the "process" is called the check-out process, and the purpose of the process is to pay for and bag your beverages. The process begins with you stepping into line, and ends with you receiving your purchases and leaving the store. You are the customer (you have the money and you have come to buy beverage), and the store is the supplier.

The process steps are the activities that you and the store personnel do to complete the transaction. In this simple example, we have described a business process. Imagine other business processes: ordering cheque booklets from banks, requesting accommodation from Hotel Presidential PH, developing new products, administering NCS conference participants registration, building a new home, etc.

Business processes are simply a set of activities that transform a set of inputs into a set of outputs (goods or services) for another person or process using people and tools. We all do them, and at one time or another play the role of customer or supplier.

You may see business processes pictured as a set of triangles as shown below. The purpose of this model is to define the supplier and process inputs, your process, and the customer and associated outputs. Also shown is the feedback loop from customers.



Why process improvement?

Improving business processes is paramount for businesses to stay competitive in today's marketplace. Over the last 10 to 15 years companies have been forced to improve their business processes because we, as customers, are demanding better and better products and services. And if we do not receive what we want from one supplier, we have many others to choose from (hence the competitive issue for businesses). Many companies began business process improvement with

Continuous Process Improvement Model

a continuous improvement model. This model attempts to understand and measure the current process, and make performance improvements accordingly.

The figure below illustrates the basic steps. You begin by documenting what you do today, establish some way to measure the process based on what your customers want, do the process, measure the results, and then identify improvement opportunities based on the data you collected. You then implement process improvements, and measure the performance of the new process. This loop repeats over and over again, and is called continuous process improvement. You might also hear it called business process improvement, functional process improvement, etc.

This method for improving business processes is effective to obtain gradual, incremental improvement. However, over the last 10 years several factors have accelerated the need to improve business processes. The most obvious is technology. New technologies (like the Internet) are rapidly bringing new capabilities to businesses, thereby raising the competitive bar and the need to improve business processes dramatically.

Another apparent trend is the opening of world markets and increased free trade. Such changes bring more companies into the marketplace, and competing becomes harder and harder. In today's marketplace, major changes are required to just stay even. It has become a matter of survival for most companies.

As a result, companies have sought out methods for faster business process improvement. Moreover, companies want breakthrough performance changes, not just incremental changes, and they want it now. Because the rate of change has increased for everyone, few businesses can afford a slow change process. One approach for rapid change and dramatic improvement that has emerged is Business Process Reengineering (BPR).

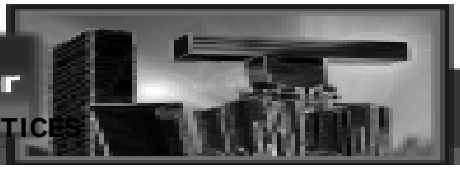
Business Process Reengineering (BPR)

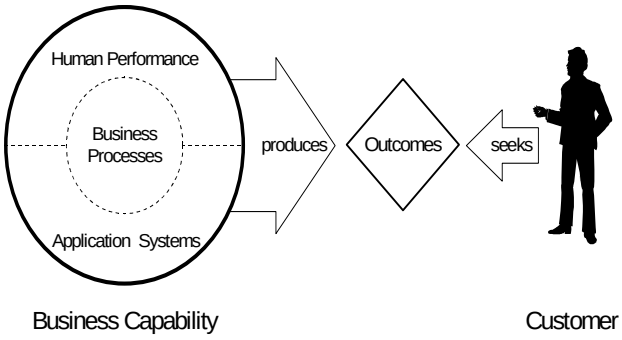
BPR relies on a different school of thought than continuous process improvement. In the extreme, reengineering assumes the current process is irrelevant - it doesn't work, it's broke, forget it. Start all over. Such a clean slate perspective enables the designers of business processes to disassociate themselves from today's process, and focus on a new process. In a manner of speaking, it is like projecting yourself into the future and asking yourself: what should the process look like? What do my customers want it to look like? What do other employees want it to look like? How do best-in-class companies do it? What might we be able to do with new technology?

Such an approach is pictured below. It begins with defining the scope and objectives of your reengineering project, then going through a learning process (with your customers, your employees, your competitors and non-competitors, and with new technology). Given this knowledge base, you can create a vision for the future and design new business processes. Given the definition of the "to be" state, you can then create a plan of action based on the gap between your current processes, technologies and structures, and where you want to go. It is then a matter of implementing your solution.

Breakthrough Reengineering Model

In summary, the extreme contrast between continuous process improvement and business process reengineering lies in where you start (with today's process, or with a clean slate), and with the magnitude and rate of resulting changes. Reengineering can be defined as a business development activity that focuses on improving the effectiveness of business processes, rather than on improving the efficiency of current operations. The focus of reengineering is a **business process:** a set of activities oriented toward assuring a given outcome. A business process always has a customer associated with it and frequently crosses traditional departmental boundaries. The business process forms the center of a **business capability**, which defines how the business will produce the desired outcome. Improving business results through improving a business capability is the goal of any reengineering effort.



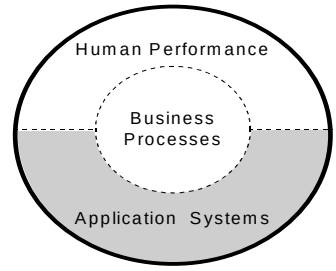


Over time many derivatives of radical, breakthrough improvement and continuous improvement have emerged that attempt to address the difficulties of implementing major change in corporations. It is difficult to find a single approach that exactly matched a particular company's needs, and the challenge is to know what method to use, when, and how to pull it off successfully such that bottom-line business results are achieved.

Reengineering involves making changes to what people do, they way they work, and the tools they use. These changes are used to improve business performance by improving business capability. Applications are an important part of a successful reengineering effort; however, for an application development effort to be successful, all personnel involved must be aware of the broader context of reengineering and must work within that context. This involves understanding what reengineering is, its impact on application development, and approaches for addressing that impact. The process of changing the business through the use of systems and applications will be the focus of this discussion.

An application can support a business process in three ways. It can provide knowledge—for example, determining a customer's credit status—through an automated portion of the business process. Applications can also provide information that enables people to make decisions, and can record the results of business processes. Finally, an application can also be responsible for automating a portion of the business process.

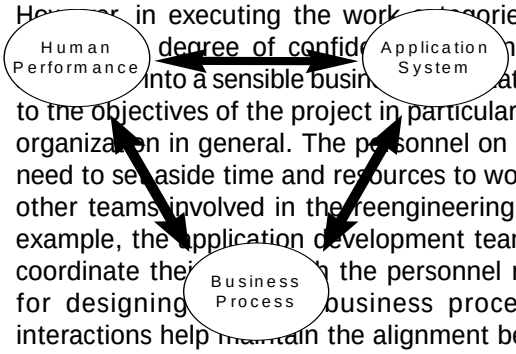
An application assists human performance by providing performance support in the form of on-line help, wizards, templates, and various navigation aids. The application development teams need to be aware of performance support requirements to ensure that they are included in the new application



Reengineering involves improving business capability. Information systems are only a part of the reengineering effort; to be successful, an application development project must be able to integrate itself within a reengineering program. That integration requires the project to recognize that reengineering scope work is broader than applications scopework.

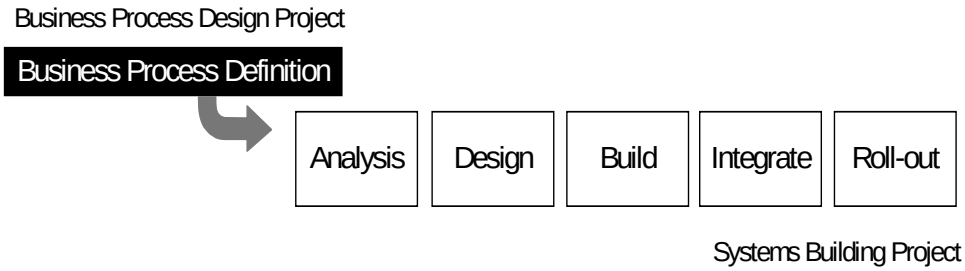
Applications are only a part of improving a business capability

Applications and Systems projects normally result from the need to improve or reengineer a business process to harness emerging opportunities.



However, in executing the work categories, there is a degree of confidence resulting into a sensible business application aligned to the objectives of the project in particular and to the organization in general. The personnel on the project need to set aside time and resources to work with the other teams involved in the reengineering effort. For example, the application development team needs to coordinate their with the personnel responsible for designing the business process. These interactions help maintain the alignment between the parts of the new business capability.

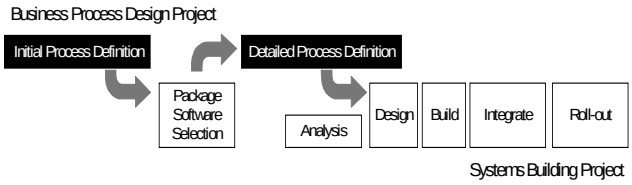
Interaction among the parts of a capability is required in reengineering



Traditional application development efforts rely on a set of business representative requirements to define scope and structure. In reengineering, these requirements are reflected primarily in a business process design that defines the application at a conceptual level. This model defines the boundaries of the application in terms of the business process. The application is therefore developed or acquired within the context of the business process.

In custom development situations, the business process definition provides the basis for the application analysis model. The application development team then implements the application in accordance with the business process. This creates a hand-off between the business process definition and application development, as shown in the figure below.

Packaged software represents an important option for implementing new systems during reengineering. It supports the reengineering effort by providing established application functionality based on a predefined business process. Packaged software selection therefore involves choosing the package with the requirements and process that best fit the needs of the reengineered business process. This is done by defining an initial business process definition to the level required to select among the various packages. The overall flow of work is shown in the figure below.



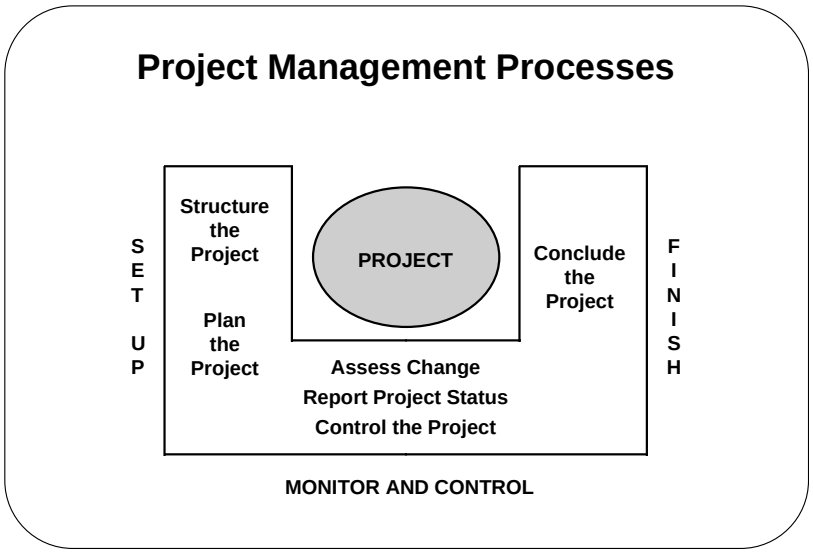
Irrespective of the business driver, and to an extent, the application delivery option, successful projects have been found to follow auditable framework shown on the picture below.

The framework can be grouped into three main categories of work viz:

- Project Management Processes and Techniques
- Project Execution Processes and Techniques and
- Project Close-Out Processes Techniques

1 Project Management Processes

Project management is the framework surrounding a project to ensure the project is brought to a successful



Structure the Project

outcome. This framework of processes constitutes the *project management life cycle* and is re-usable for any project.

Some of these processes (*Structure the Project*, *Plan the Project*, and *Conclude the Project*) are used at specific points in the project. Other processes (*Assess Change, Report Project Status*, and *Control the Project*) are more event-driven and are used continuously throughout the project.

The *Structure the Project* process is used to develop the initial definition of a project. The aims of this process are: to identify and document the business objectives that the project will support; to establish sponsorship for the project; to define the objectives, approach, and controls of the project; and to estimate the project's effort, duration, and cost. For some organizations, this first project document, Terms of Reference (sometimes called the project prospectus) is the definition document or Business Case.

To structure a project, the project manager selects the appropriate route to take, determines the deliverables that will be created, and develops a high-level workplan, estimates, and budget. In addition, key strategies for acquisition/development, quality management, risk management, issue management, scope management, and training are defined. All these components are contained in the Project Charter and should be completed to a sufficient level to gain sponsor approval of the project.

Plan the Project

The *Plan the Project* process refines the project charter and creates various detailed project plans, which comprise the operational plan that is used in the day-to-day management and control of the project. These plans should include task schedules, resource schedules, a quality plan, a risk management plan, an issue management plan, a scope management plan, a knowledge co-ordination plan, and a project budget. This effort normally occurs during the start-up stage of the project, but this process is used to revise the detailed project plans as necessary throughout the life of the project.

Assess Change

Change is defined as an addition to, deletion from, or modification of project scope as it is defined in the project charter. The *Assess Change* process involves the documentation, evaluation, and disposition of change requests arising from sources internal or external to the project team. The project scope is determined by the content of the project charter and all approved change requests. If significant changes to the project scope are encountered, the project charter should be updated.

Report Project Status

The *Report Project Status* process communicates project progress against project plans. Project status is communicated to members of the project management and control structure, the user community, and other interested parties. Information on project status is provided by the *Control the Project* process. The *Report Project Status* process is invoked periodically on a formal basis and as required on a less formal basis.

Control the Project

The *Control the Project* process monitors and evaluates progress of the project against project plans and initiates corrective actions. As such, it is a continuing process. Control is applied periodically on a formal basis and as required on an informal basis. One primary focus of the *Control the Project* process is damage control - avoiding damaging, uncontrollable circumstances and limiting their effects.

Conclude the Project

The *Conclude the Project* process normally occurs in the final stage of a project. It takes place only after members of the project management and control structure have agreed that the project is to be terminated. Termination is intended to be the successful completion of the project in achieving its objectives according to the standards laid down in the project charter. However, termination may also be the controlled conclusion of a project which is to proceed no further, for whatever reasons. It is the process by which the project is formally concluded, and the project and its resources formally evaluated. As such, it occurs only once during a project.

A key objective of the *Conclude the Project* process is to augment the historical base of project information, and to provide information for improvement of the project management processes. This base is used to improve the metrics (e.g. estimating guidelines) for future projects and to identify reusable deliverables.

2 Project Execution Processes

Prior to this stage, the necessary documentations (TOR, Project Charter and Plan) would have been completed. Also required to have been completed are the Business Area Analysis, Solution Definition, Conceptual Design, Entity Relation Diagrams and where necessary the Algorithms and the Database size estimates (initial disk storage and expected annual growth rate). A decision on the implementation strategy would also have been taken based on the maturity of the application area. This will help if things go wrong during the project.

The work can then be executed in measured steps as outlined below

- Set up the application development/delivery environment
- If data is to be shared in the application, then set up the application's database
- Create and compile the application source modules (the basic units and logical groups)
- Create test data in the database
- Debug the basic units (the Functions and Sub-procedures)
- Carry out unit testing of the basic units and integration testing of logical groupings of the basic units (the program modules)
- Build an installation script for use to install and integrate all modules and components
- Debug and Test the installation script
- Run the installation script to install the whole application
- Make ready for System and User Acceptance testing of the application ensuring the completion of the four levels:

  (1) Unit testing - the testing of basic units (e.g. Functions and Sub-routines) to ensure that they are working according to design specification

  (2) Integration testing - the progressive testing of major components or modules, which are logical groupings of the basic units that have been unit-tested, to ensure that the basic units work harmoniously together and properly and the major components work as designed

  (3) System testing - the testing of all the components of the application as a whole in its final form (as it will be in production mode) and the purpose is to ascertain that it meets all the requirements of the system.

  (4) User Acceptance testing - like the System testing but done by user representatives to certify that their requirements are met and therefore formally accept the system.

3 Project Close-Out Processes

After the testing and acceptance procedure is completed, the applications shall pass through the full operational acceptance procedures to implement it for operational use. The implementation process consists of **quality assurance and control** (QA/QC) and **transfer to production** which includes the **change management** procedure (user training and roll-out). The purpose of the QA/QC procedure is to ensure that the application is of acceptable quality in terms of meeting business, technical and operational requirements. The transfer of the application to production environment must follow the QA/QC procedure for a full operational acceptance. The purpose of the transfer is to install the application and its database in the production environment in such a way that it fits in to the environment without adversely affecting other existing applications and services. Transfer to production involves the following steps:

- Initiate change management procedure for the purpose of installing the application and its database in the production environment.
- Execute the change management procedure to install the application and its database in the production environment. Ensure that all stakeholders are adequately informed before any installation begins and it is recommended that installation be done outside prime time
- Include this application in the standard house keeping procedures
- Check-in the source code of developed modules and other available source files into the source code library in the production environment
- Register this application in the inventory of existing applications
- Hand over to the Application Support and Operations teams and sign-off Agreement-to-operate.

4 Techniques Building Synergistic Teams

Projects are better delivered by a team of people. There must therefore be a clause that should include the right to monitor the performance of individuals in the team and to change non performers. Team membership should include both IT professionals and the owners of the business process requiring changes. Therefore it is

advisable to define clear roles, responsibilities and where authority lies. Encourage communication with team members on team and individual basis to help them perform well. It can be an incentive to team members if they know that they will be trained to improve their skills.

Defining the Project Organisation

Resources can be obtained from a number of different sources, such as users, IT department or third-parties (e.g. software houses, sub-contractors, independent consultants). It is advisable to consider all the resourcing options. The aim is to ensure that appropriate type and level of resources and skills are available throughout the life of the project.

The level of involvement of different groups will change as the project progresses. For example, users may dominate in the some phases (e.g. specification), whilst IT involvement will be greater in other phases (e.g. construction). Roles and responsibilities can be used to identify and assess the resources and competences required from the team.

Developing a Project Charter

The project charter is the "contract" between the project implementation team and the project sponsor. The charter can be used as the basis of the agreement with sponsor, because it:

- Provides a clear statement of the purpose of the project and what the team is committed to deliver,
- Defines the deliverables, broken down by components, that must be produced,
- Defines the project roles and responsibilities,
- Defines the risks and the strategies to minimise them,
- Makes visible the development process and the approach that will be used to manage the project,
- Establishes the ground rules for the project,
- Provides a baseline for scope and expectation management,
- Provides the foundation for preparing detailed plans to be used throughout the life of the project.

Developing a Project Workplan

The project workplan is the basis for measuring the progress of the project and the performance of the team. It should make visible all project-related work, not just the development tasks. Include administration tasks to cover scope management, issue management, risk management, reporting, etc. Also, schedule regular progress meetings with the sponsor and formal reviews of deliverables as an integral part of the project, not a peripheral activity.

Establishing Scope

A clear, agreed scope of the project is fundamental to a successful completion. Don't skimp on this. The scope of the project can still be made clear even if one is unsure of the specification of the system. Establish the strategy so that additional risks associated with a loose specification are covered.

Estimating the Work

Make an internal estimate of the work to be done. Estimate the work from different perspectives to provide a level of confidence in the figures. For example, an estimate based on a work breakdown structure can be checked against an estimate from a Function Point count or a commercial database. Don't rely on an estimate from just one perspective.

Managing Issues

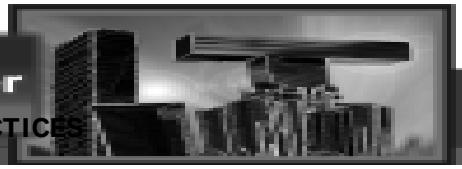
Issues arising have to be dealt with in a timely manner. Have the issue management procedures in place before starting the work. Specify escalation procedures and responsibilities for when agreement cannot be reached at the project working level.

Managing Quality

Use of knowledgeable staff in the project will lead to improved quality of delivered applications as they increases their understanding of the business over time. If several people are involved in the project, include walkthroughs of the interfacing between the different tasks with all parties present. Don't allow one member to change anything without letting the others know.

Managing Risk

The work activities should be geared to the risks involved in the project. For example, if prototyping is to be done, ensure you can review the project if the prototype is unsuccessful. Regularly review the risks as the project progresses, for example at the initiation of an issue or change request. You should also re-evaluate risks at the delivery of each major deliverable.



Be prepared to review the project if the re-evaluated risks are unacceptable.

Managing Scope

Changes to scope are often allowed. Have change control procedures in place before starting the work, and enforce them rigorously. Make sure that the scope change procedures are agreed with the sponsor.

Managing User Expectations

Users may be more remote from the project. Use the project charter to set users' expectations, and project status reporting to manage expectations. Educate the users on the importance of change and respond to issues timely.

Monitoring Activity

Deliverables form the basis of the project. Ensure that all deliverables are produced to the agreed specification and timing. Any attempt to combine deliverables or delay delivery could be treated as non-compliance.

Use regular meetings of the steering committee and project team and the inspection of deliverables to monitor progress and flush out potential problems before they occur. Problems should be discussed and resolved but if not possible then record the item on the issues log and on the agenda for the steering committee.

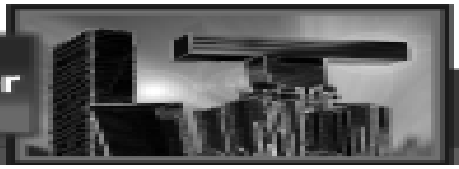
Reporting Project Progress

Adopt standard forms for reporting progress. The forms should capture any required company-specific metrics. Report progress using the forms.

References

Microsoft Solution Framework, Microsoft Inc.

Stage Gate Framework, Shell



System Integration: Towards ATM Networks Implementation in Nigeria

W. N. Ekemezie, E. U. Ezeorah

ABSTRACT

The implementation of a bank's consumer networks is often complicated by the unusual degree of reliability needed. One of such networks, an n-terminal Automated Teller Machine (ATM) network that can be installed at its branch locations, while supervising its control at the head office, typifies the extent of the measures taken to ensure this reliability. The number n could range from 10 to 200 or higher, depending on the strength of the establishing firm. The day-to-day financial transactions and other related issues carried out by Nigerian banks, insurance firms, schools, seaports, airports, road and rail transport systems require such cash dispensers as Community Electronic Teller System (COMETS), Customer Account Management System (CAMS), Electronic Cash Register (ECR), and/or the 24-hour ATM network to facilitate the problem of handling visa accounts, authorizing and verifying credit cards, bank-by-phone system, student loans operations, and cashing cheques/withdrawal slips. The key to an ATM network is the degree to which a bank and its customers can trust it to be accurate and dependable. In the light of the above, this paper discusses the implementation of an ATM network under the following headings:- network configuration, tracing a transaction through a network, reliability guaranteed, social and economic benefits of ATM network application in Nigeria.

Keywords: Configuration, Reliability, Challenges, and Benefits.

Introduction

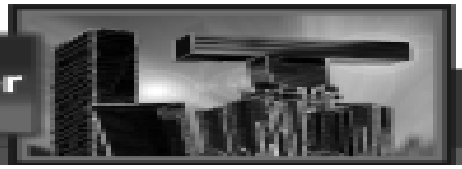
Aside the unquantifiable man-hours lost in queuing up in banking halls, is the cost of adopting modern tools and processes in delivering banking services. Changing the way financial services are used in international trade, local trade and individual needs, according to Jain (1996), electronic finance is forcing standardization, adding speed and reducing costs. These changes can help developing countries (Nigeria and others) improve their international competitiveness.

For any bank not to be left behind in the race to achieve multi-branch banking and deliver faster, with more innovative services, it must have to install the most modern systems available. These include the banking software, computers with high speed processors and large memories and the right personnel. For most banks, the right mix has not been achieved, and that is why return on investment has not been as high as many expected. Perhaps, that partly explains why some of them are going close to bankruptcy. According to Odubele (2003), flex-cub is gradually penetrating banking applications, owing to the need to have different delivering channels for the customers and also making banking business convenient to them. With flex-cube, one can have different delivering channels such as ATMs, POS, Tele-banking, Internet banking, etc. Arise (2004), in his statistics of cash dispensers world wide showed that there are about 900,000 ATM locations, 32.8 million merchants' acceptance locations in over 30,000 financial institutions across six continents.

The CBN's sledge hammer of bank's capitalization base of N25 billion, by 31st December, 2006, has forced many financial institutions, particularly banks, to devise means of meeting the target by merging or attracting different categories of customers to beef up their transactions. The new generation banks rely on the latter. ATMs are employed as a means of meeting the e-commerce and e-payment strategies, which are the order of the day in developed economies. With this new technology, customers other than Nigerians are attracted.

System Requirements

The key to an ATM network is its degree of dependence and accuracy to which a bank and its customers can trust. In other words, its acceptance depends heavily on two considerations: ease of operation and high



reliability. A network offering the greatest possible ease of operation would hardly be received in the absence of its reliability. On this basis, the following devices are inescapable in ATM configuration:

- ATM controllers (software and hardware)
- ATM machines
- Host computer
- Front-End-Processor (FEP)
- Data communication links running at 2.4Kbits/s. These interface directly with customers.
- Modems Sharing Devices (MSDs)
- Modems (dedicated types) running at 1.2Kbits/s
- Telephone lines, which provides dial-up-line backup
- EIA RS-232-C switches
- Remote Transfer Switches (RTS) running at 9.6Kbits/s
- Communication Management Systems (CMS) modular switches, which transmits financial data to and from the Host computer via the FEP. Each of these is attached with two modems running at 1.2Kbits/s.
- Multi-node Telecom Bridges, which are interfaced with the ATM controllers.
- Remote Digital Mixing Modules
- Network Diagnostic Controllers, which serve as the main channels running at higher frequencies than the diagnostic channels in EIA RS-232-C switch.
- Diagnostic Channels, which serve as the secondary channels running at lower frequencies than the NDCs thus, avoiding data collision with the main channel during commands.

Fig. 2: Telco Bridge Table

Address	Port Number
BOF61A1	4
BOF61A2	2
BOF61A3	4
BOF61A4	2
BOF61A5	4
BOF61A6	2
BOF61A7	4
BOF61A8	2
BOF61A9	3
BOF61A10	1
…	

The ATM Network Configuration

Significance to network success is alternate-path switching, which places an ATM back on-line if a modem or line failure occurs. In alternate-path switching, if a phone line or a modem should fail, the ATM continues to operate on alternate communications line of the second ATM at the site. The second ATM's modem would act as a backup and provide data to both ATMs via the built-in Modem Sharing Devices (Fig. 3).

It is possible that components added to a network to achieve higher reliability might have a negative impact. The reliability of the backup hardware and the techniques used to implement this hardware must not degrade the overall network. All backup equipment must be monitored and tested to verify that it will operate properly. The control unit of the CMS switches provides an integral part of the diagnostic network (Fig. 3). This control unit has it own address, just as the diagnostic channel of each data modem has a separate address. This is discussed further in the preceding paragraph.

A command sent from the Network Diagnostic Controller (NDC), at the central site, to the MSD by way of the remaining good path would cause the modular switch to connect the inaccessible ATM to the active modem through integrated MSDs. This is accomplished by the CMS (Fig.3), which consists of standard EIA RS-232-C switches, two MSDs, and a microprocessor control card. A pair of ATMs at each branch site is serviced by one of these switches. The switching function is initiated by a command from the NDC located at the Head Office (A) and the Sub-Controller site (B). In addition to switching, the device checks itself, providing status information concerning its own operations whenever a problem is detected. This device acts as the main channel.

The secondary channel in the EIA RS-232-C, which carries asynchronous data at 75bits/s, perform additional functions besides monitoring, testing and verifying communications operations. These include remote switch controlling, and monitoring. To avoid data 'jam' with signals from the main channel during commands, it transmits at a lower frequency than the primary channel. The two channels provide the overall systems' capabilities.

Controllers at the Head Office (Host-site) are linked to the Front-End Processor (FEP) by communications links - ComLink III short-haul modems running at 2.4kbit/s and MSDs (Fig. 1A). Similarly, the controllers at the branch site are site linked to the FEP by way of a multi-node modem operating at a speed of 9.6kbit/s (for the four EIA ports running at 2.4kbit/s each) and by MSDs.

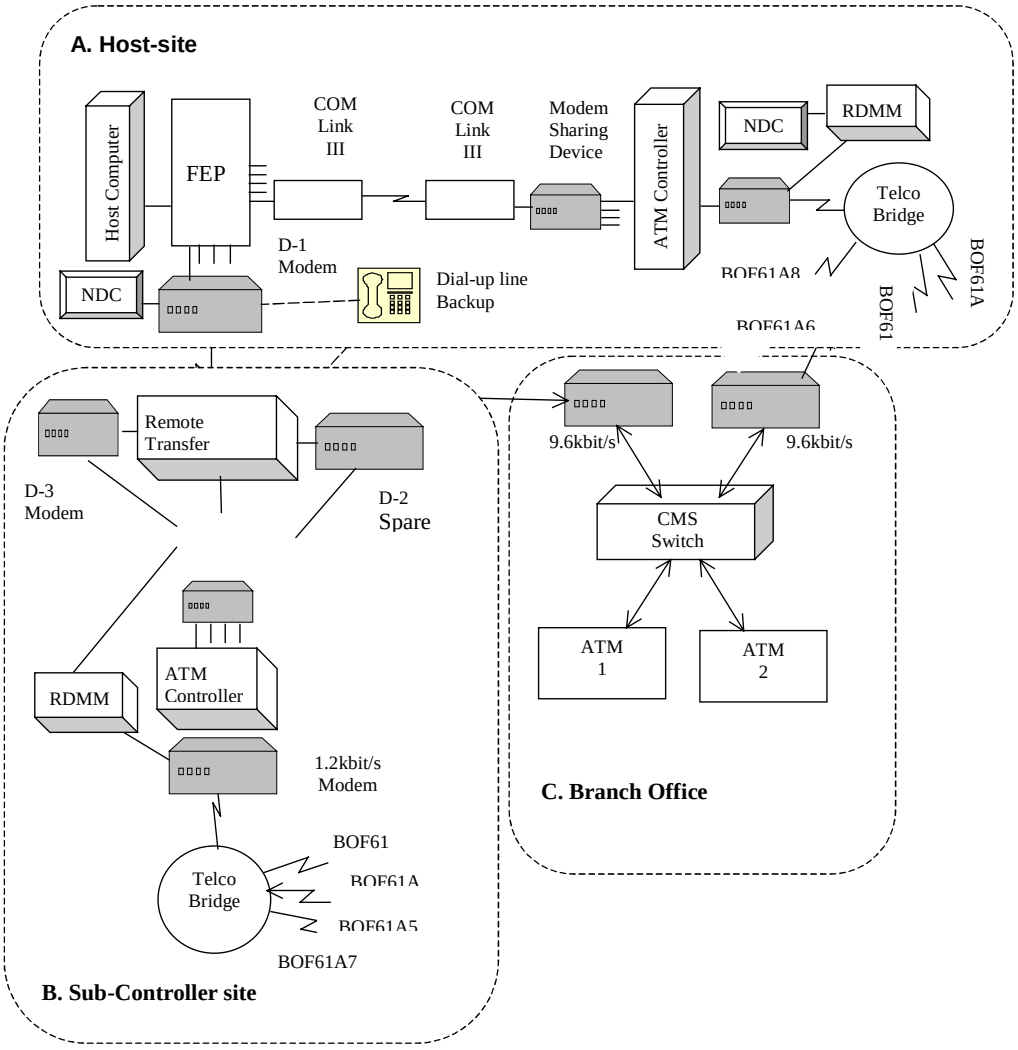


Fig. 1 Bird's-eye view of the Network. To ensure redundancy each branch has at least two ATMs (C). One link provides the connection to the host-site (A) via dedicated lines. The other link leads to the controller at sub-head office (B), which in turn is connected to the host via a 9.6kbit/s line with dial-up backup capabilities.

Special Functions

The microprocessor control; card is polled by the diagnostic channel as part of the constant network polling (Fig. 3). Failure of the switch control to respond to the poll causes an alarm at the central site. At this point, the operator performs fault isolation to determine the nature of the problem and then enforces the appropriate solution. The switch control card has two control input paths to ensure operation in the event one control path should fail.

All remote-site switching is controlled from the central-site diagnostic network control panels. Switches are controlled on the same CRT that is used to monitor the communications network. This eliminates the need for separate control panels or CRT terminals to perform the diagnostic functions.

Each modem has input and output controls, which connect the modem elements for both normal and test-mode configurations. Its interruptive diagnostics include self-test, end-to-end, line-loop, digital loop-back, and error tests. Automatic fault-reporting provides alarm for modem power failure, streaming (constant carrier in a switched-carrier environment), and receive-line failure. Additional diagnostics include line not restored, loss of synchronism (the FEP's framing pattern), terminal loop faults, and external user provided alarm.

Telco Bridge

The computer at the host-site sends binary data to the modulator portion of the modem through the controller. The data is sent as an analog signal on the telephone

lines. The modem at the Branch Office can also send data to the computer at the host-site using the same modulation/demodulation processes.

The Telco Bridge, which maintains a table (Fig. 2), checks the physical (medium access control) addresses (source and destination) contained in the frame. It decides if the frame should be forwarded or dropped. If the frame is to be forwarded, the decision must specify the port number. The bridge table mentioned earlier maps addresses to ports.

If a frame is destined for station Branch Office 61A3, and arrives at port#1, the bridge consults its table to find the departing port. According to its table, frames for Branch Office 61A3 leave through port#1; therefore, there is no need forwarding; the frame is dropped. On the other hand, if a frame for Branch Office 61A4 arrives at port#1, the departing port is port#2 and the frame is forwarded.

Tracing A Transaction

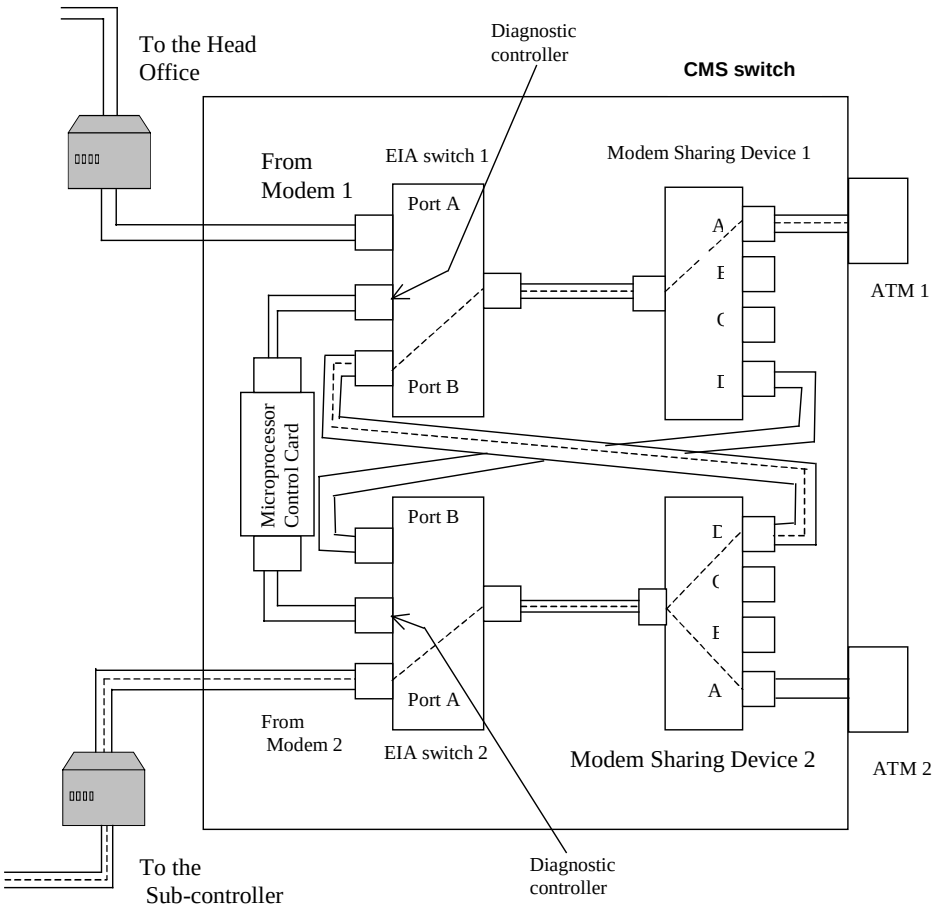
When an ATM customer inserts a bank-issued card into an ATM at the bank's branch, for example, the ATM requests service of the host computer via its communications link to the ATM controller at the head office, which passes it on to the front-end processor (Fig.1). The other ATM has its requests for data routed over its 1.2kbit/s link and on to the controller site through a 9.6 Kbit/s link.

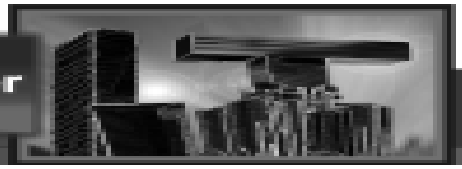
At the point of failure, the operator can monitor all the ATMs, by pinpointing those that are low on cash, so that plans can be made for replenishment in the event that a particular ATM is low in cash. The operator can also tell which ATM has a problem with the help of the COM links. If the 9.6 Kbit/s private line fails, an alternative link can be established using the dial-up line backup configuration. One diagnostic controller backs up the other in detecting line outages and modem problems.

The complete Arrangement

By incorporating network control, the network becomes one of shared information management that allows not only for the transfer of financial data but also for

Fig. 3 CMS switch, which shows ATM 1 in a backup mode, relying on the diagnostic controls. These transmit at a lower frequency than the main channel.





advanced diagnostic and control functions.

The present-day advancement of this concept is among the most important factors in future network advancement and growth, as well as return on banking investments. The continuing expansion on data communications network operations illustrates the need of allowing central-site technicians to know where and when a failure has occurred, as well as the specific element that has failed (such as a circuit, terminal, modem, or multiplexer). The backup equipment and techniques enable operators to bypass the failing component.

This procedure permits the operator to accurately dispatch either internal or vendor personnel to restore service. The bank then has a tool for managing not only its networks but also its service organizations, regardless of site location. To achieve overall network economies, management and control must be maintained at all times.

Challenges of ATM Implementation in Nigeria

These include:

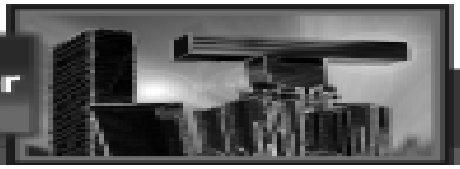
- Absence of telephones to the larger population.
- Inconsistent supply of electricity either as source of power or for recharging the battery-powered ones. Most of the rural areas, where urban dwellers can make withdrawals while at their various homes, have no electricity supply.
- Absence of Electronic Fund Transfer (EFT) and processing network that would facilitate transaction routing between ATMs and banks.
- High cost of acquiring, deploying and operating ATMs.
- High telecommunications tariffs, and poor telecoms infrastructure.

Benefits

- Banks offer and manage profitable banking services without the presence of the banks' personnel (overtime costs are saved).
- Maintains a bank account at very low operational costs.
- Affords processing of small transactions that are typical to the target market.
- Integrates new payment services to low income customers sharing a common banking infrastructure and service base.
- Provides savings via its ubiquitous banking services.
- Provides savings from the cost of moving and storing cash, which substantially reduce overhead cost; thereby yielding profits to the banks.
- Permits interoperability. That is, when a user inserts his card issued by Bank 'A' in an ATM meant for Bank 'B', the ATM will act as if it belongs to Bank 'A'.
- Improves customer service delivery.
- Creates avenues for new income, in terms of fees and charges on their use, for banks.
- The associated card (MasterCard) will help alleviate the burden of carrying cash, traveler's cheques and ease many other social problems Nigerians face when they travel outside the country. For instance, hotel reservations can be made at the comfort of one's home or nearby cyber-cafe and amount paid instantly, assuming a sudden invitation or visit. The mode of checking into hotels worldwide has now made credit cards almost a must for all guests.
- Enables Nigerians to truly participate in electronic commerce, in which case, transactions can be initiated and concluded instantly for goods ordered from Europe, America, Asia, etc., and shipped via DHL, FedEx, UPS, etc., without necessarily incurring the cost of a travel ticket.
- Eliminates cash-induced robbery in our society, since there is no need for anyone to carry loads of cash either to or from any country as such funds can always be kept in the smartcard.
- With MasterCard, Departments; Faculties; or students can order books online, subscribe to journals, borrow books through e-Libraries worldwide and make payment through a reliable merchant.
- Parents/Guardians can pay their wards' fees outside the country using the cards. While progressing in their studies, these students would be frustrated, if they are ignorant of these cards, since they would use them to perform one essential activity or the other.

Areas of Application

- Banking sector.
- Telecommunications sector.
- Government: National Health Insurance Scheme



(NHIS), Poverty Alleviation Scheme (PAS), Retirement Payment Scheme (RPS), ECOWAS Travel Scheme (ETS), and the Salary Payment System (SPS), etc.

- Students Scheme: A typical multi-application card could contain information on personal details, authentication of student identity, electronic cash, library and attendance monitoring. When a student card is tied to a bank, it is used in ATM machines and at POS terminals.
- Petroleum sector: As in petrol station for fuel, lubricants and snacks purchases.
- Transportation sector (Toll plazas): The banks can ingrate this to their cards solution whereby debits can be made automatically from the holder's account.
- Employee Payroll: The cards can act as ID cards containing all bio-data. It will also act as credit or mostly debit cards and the employee only needs to access his/her salary on payday without waiting for any paymaster or payslip.
- The Government, the employers and the banks will do this jointly. The technology ensures that Card Acceptor Terminals (CATs) communicate with the switch and the switch is able verify funds availability.
- Distribution sector: E.g., Nigeria Breweries and their clients (suppliers and recipients of raw materials and finished products respectively).
- Merchants' shops and parks.
- Internet sector.

Banks Offering ATM Services in Nigeria

- Ecobank Plc. In addition, this bank issues MasterCard.
- Union Bank of Nigeria Plc
- First Bank of Nigeria Plc
- Societe Generale Bank of Nigeria Plc
- Guarantee Trust Bank Plc
- International Bank Plc
- United Bank Africa Plc
- Standard Trust Bank Plc
- Universal Trust Bank Plc

Other Cash Dispensers

1. Community Electronic Teller System (COMETS): Serves as a network centre for general branch banking operations. Depending on its capability, it can support upwards of 1,800 terminals.
2. Customer Account Management System (CAMS): Handles all aspects of visa accounts of a bank's employees and interested customers. The network can support 400 terminals.
3. Electronic Cash Register (ECR): Checks credit card authorization and verification.
4. Point-of-Sale (POS) terminal
5. Other networks can be developed to handle
   i. Bank-by-phone operation
   ii. Student Loans operations

Conclusion

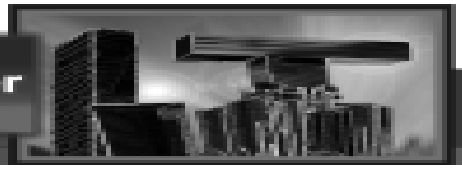
Banking services have moved from the traditional method, (or the one aided by technology) to one that can be done from the comfort of one's home. The ancillary infrastructure, such as good telecommunications, better Internet services, achieving greater penetration, and rising level of awareness among the banking public that they do not need to physically interact with the teller in his bank for him to achieve his cash needs.

It is important that banks recognize the need for some form of collaborative effort that would promote further development and growth of the ATM industry and electronic commerce in Nigeria. Adopting an ISO approach would no doubt be the key to achieving growth in the ATM industry. ATM deployment should be approached from a long-term perspective and not the usual lackadaisical attitude of some staff, typical of most Nigerian banks. Baring the prevailing economic difficulties, a consortium of banks or private initiatives should brace up and establish ATM networks in order to embrace the metropolitan, urban, cities, and most importantly the local community dwellers. It is obvious that both foreign and local banks will patronize such outfits.

References

Ayo, Arise (2004) "MasterCard in the Hands of Nigerians" e-Business Perspective, THISDAY, Vol. 10, No. 3482.

Behrouz, A. F. (2004) **Data Communications and Networking** Third Edition, Tata



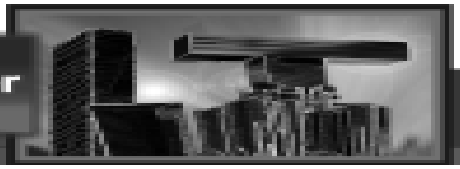
McGraw-Hill Publishing, New Delhi. p. 129.

Jain, R. "Congestion Control in Computer Networks: Issues and Trends" IEEE

Network, May 1990 pp. 24-30

Odubele, J. "Using ICT to Help Organizations", Daily Independent online, Daily Independent Newspapers, January 6, 2003. Source from: http://www.dailyindependentonline.com.

Tritto, Charles (1982) "Automation of Chase Manhattan Bank" N.A. New Hude Park, N.Y.



Controls for Preventing and Detection of Computer Crime

Peter O. Olayiwola

A B S T R A C T

It would be nice to assume that everyone associated with a business is honest. That assumption would certainly eliminate the need for crime prevention, but, of course, it is not viable. People will commit crimes for many reasons, some of which are rational, others of which may make no sense to the observer. The larger the organization, the more likely it is that someone is out to commit a crime. Managers who subscribe to this belief are not necessarily paranoid. The factors that enhance the probability that an organisation will be the target of theft, fraud, embezzlement and corruption, including computer crime, can be either motivational (related to the corporate reward system and company policies) or personal (related to the character of a particular perpetrator). These factors will be discussed in details. Internal control and security systems are designed on the basis of past experience, both in the company in which they are installed and in other companies. The challenge here is to build in enough controls to discourage and catch criminal behaviour without breaking the bank in costs or going overboard on security.mmComputer crime can be discouraged through measures designed not only to prevent crime but also to detect attempts to engage in computer crimes. The recommended prevention measures will be adequately discussed in the paper. The focus of this paper is the specific measures that are often used to prevent computer crimes by those either inside or outside an organization. Measures to detect attempts to commit computer crime will be also be discussed.

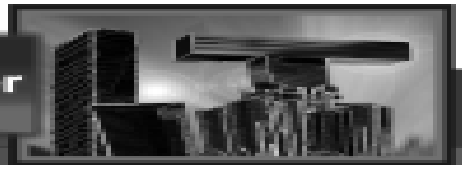
1.0 INTRODUCTION

It would be so wonderful to assume that everyone associated with a business is honest. That assumption would certainly eliminate the need for crime prevention, but, of course, it would be a naïve assumption. People will commit crimes for many reasons, some of which are rational, others of which may make no sense to the observer. The larger the organization, the more likely it is that someone is out to commit a crime. Managers who subscribe to this belief are not necessarily paranoid. In fact, experience has shown that a lot of employees who were once highly trusted by their employers have betrayed that trust. According to the Association of Certified Fraud Examiners' (ACFE) 2004 Report to the Nation on Occupational Fraud and Abuse (p.iii), a study of 508 organizations by the ACFE reported a total of over $761 million in losses. According to the same study, a typical organization losses an estimated 6% of its total revenue to fraud. Hence, this paper will focus on both the prevention and detection of computer crime.

2.0 INTERNAL CONTROL AND SECURITY SYSTEMS

Internal control and security systems are designed on the basis of past experience, both in the organization in which they are installed and in other organizations. The challenge here is to build in enough controls to discourage and catch criminal behaviour without breaking the bank in costs or going overboard on security. Organizations that have been victimized often react by increasing controls to the point at which the controls can become oppressive and actually interfere with the organization's operations. Although it is rational for organizations to set up rules to define acceptable and unacceptable behavior, too many constraints make people feel oppressed, distrusted, and become self-conscious of being under constant surveillance.

Well-designed controls should provide checks and balances. We need to consider the risks, threats, and other vulnerabilities in modern society and its technological environment while simultaneously taking into account our responsibilities to employees, the value



of their contributions, and their need for satisfaction in their workplace. This includes the provision of a work environment that encourages outstanding performance, profitability and efficiency.

A competent systems analyst or information security specialist can design layer upon layer of controls. However, controls in excess of those required by the nature of the risks involved are not cost-effective and can place undue burdens both on those who must work under them and those who must monitor and control them. An organization's requirement for effective internal-controls does not represent a justification for a siege mentality or the construction of an impregnable fortress. Done effectively, the development of internal controls is a matter of proper balance and equilibrium and should not create paranoia.

Looking at the potential for theft and fraud and the actions available to prevent crime brings forth several conclusions:

- Most prevention efforts focus on building more accounting, access or physical security controls.
- It is vital to recognize that there are limits to technological and procedural controls. Given the speed with which computer and data communications technology evolves and the complexity of modern systems, it is difficult for improvements in protection and detection mechanisms to keep pace.
- It is also important for organizations to recognize that improvements in the working environment, including a positive ethical climate and strong interpersonal trust, help to discourage criminal thinking and behaviour and, as a result, are a part of the control environment. Some factors in the business environment are likely to encourage computer crime while others discourage crime. Clearly, we want to minimize the criminal behavioural motivators and maximize the non-criminal motivators.

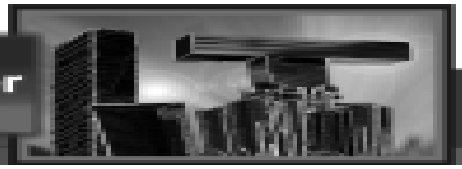
3.0 FACTORS ENCOURAGING COMPUTER CRIME

The factors that enhance the probability that an organization will be the target of theft, fraud, embezzlement and corruption, including computer crime, can be either motivational (related to the corporate reward system and organization policies) or personal (related to the character of a particular perpetrator of the crime).

3.1 MOTIVATIONAL FACTORS ENCOURAGING COMPUTER CRIME

The following are motivational factors that encourage computer crime:

- Inadequate management controls, including failure to communicate expected standards of job-related performance or on-the-job behaviour, ambiguity in work roles, relationships, responsibilities and areas of accountability
- Inadequate rewards, including pay, fringe benefits, bonuses, incentives, perquisites, job security, meaningful work and promotional opportunities
- Failure to offer counseling when performance or behaviour falls below acceptable levels
- Acceptance of mediocre performance as the standard
- Inadequate reinforcement and performance feedback mechanisms, including lack of recognition for good work, loyalty, longevity and effort; lack of meaningful recognition for outstanding performance; delayed or nonexistent feedback on performance inadequacies or unacceptable on-the-job behaviour
- Inadequate support and lack of resources to meet standards by, for example, not providing authority to hire sufficient personnel to meet requirements for quality, quantity, and timeliness of work produced
- Failure to control bias or unfairness in selection, promotion, compensation, and appraisal
- An uncertain future (where a company faces merger, acquisition or failure, people may feel justified in "watching out for themselves.")
- Condoning inappropriate ethical norms or inappropriate behaviour
- Failure to control hostility generated by promotion or destructive competitiveness among departments, offices, or personnel
- Inadequate operational reviews, audits, inspections, and follow-ups to assure compliance with organizational policies,



priorities, procedures, and government regulations.

3.2 PERSONNEL-BASED FACTORS

The following are personal or personnel-based factors encouraging computer crime:

- Unresolved personal financial problems
- Inadequate orientation and training on security matters and on sanctions for violating security rules
- Inadequate standards of recruitment and selection
- Failure to screen and check the background personnel before appointing such persons to sensitive positions (This includes verification of prior employment, verification of educational qualifications, verification of financial stability and examination of character.)
- Inadequate control of the level of job-related stress and anxiety
- Unresolved problems relating to personal status
- Inadequate employee communication programs to control, to the extent possible, uncertainty and anxiety among employees

4.0 FACTORS FOR THE DETERRENCE OF COMPUTER CRIME

Computer crime can be discouraged through measures designed not only to prevent crime but also to detect attempts to engage in computer crimes. The recommended prevention measures are the following:

4.1 INTERNAL ACCOUNTING CONTROLS

These are the traditional measures that discourage crime, and they are as important in an automated environment as in a manual-processing environment. These include:

4.1.1 Separation and rotation of duties (Remember that as personnel change jobs, it is vital to update the list of computer applications that they can access so that their access at any given time matches their current job requirements.)

4.1.2 Establishment of dual signature authorities, dollar authorization limits, expiration dates for signature authorizations and check amount limits (These authorities also should be examined on both a routine and surprise basis.)

4.1.3 Offline controls and limits, including batch controls and hash totals Periodic internal audits, surprise inspections and computer security reviews

4.1.4 Absolute insistence that control policies and procedures be documented in writing.

4.2 COMPUTER ACCESS CONTROLS

These controls may include the following:

4.2.1 Authentication and identification controls, including keys or smartcards, passwords, biometrics, callback systems, one-time passwords, time and day constrained access and periodic code and password changes

4.2.2 Compartmentalization, also known as “need to know”

4.2.3 Use of encryption to protect data while stored or in transit.

4.3 FIREWALLS AND ANTIVIRUS SOLUTIONS

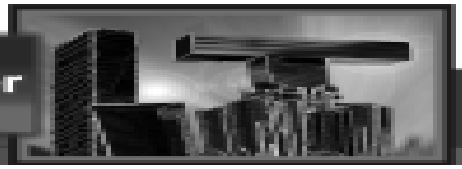
The use of firewalls and similar safeguards prevents unauthorized access and destructive attacks through the Internet.

5.0 MEASURES TO DETECT ATTEMPTS TO COMMIT COMPUTER CRIME

In this age of computer espionage and highly sophisticated computer-based criminal activities, it is important for ICT practitioners to develop the skills and detecting attempts to commit computer crime. Some of the methods for detecting computer crime include the following:

5.1 A system of logging and follow-up of exceptions should be designed and implemented to log unusual activities. Procedures should be in place to follow up on reported exceptions, such as the following:

- Transactions that are out of sequence, out of priority, or otherwise out-of-standard



- Aborted runs and entries, including repeated unsuccessful attempts to enter the system
- Attempts to access applications or functions beyond a person's authorization.

5.2 Logging and following up on variances should be able to indicate a problem may have occurred or is occurring.

5.3 General logging should be in place because, if problems are uncovered, logs of access, web activity, etc., may be vital evidence in tracking down the person involved. Logs should be maintained for at least a few months before being erased.

5.4 Use newly developed intrusion detection systems that use artificial intelligence capabilities to detect unusual transactions flowing through a system.

5.5 Develop and maintain awareness of employee attitudes and satisfaction levels.

5.6 Develop and maintain sensitivity to reports that particular individuals are having problems, living beyond their means, or talking about "getting even," for perceived slights

6.0 SECURITY MEASURES FOR PREVENTING COMPUTER CRIME

This section focuses on the specific measures that are useful for the prevention of computer crimes. Although some measures are applicable to almost all situations, it is vital that each organization considers those controls that are appropriate to its own particular circumstances.

6.1 MONITORING AND PROMPT CLOSURE OF SECURITY HOLES

One of the unpleasant realities of today's systems environment is that the systems we use, including operating systems, firewalls, and security packages, as well as application systems, may not be perfect when they are released by the manufacturers to their customers. Security holes and problems are discovered regularly. Information about ways to exploit security holes is quickly reported worldwide by independent bulletin board systems, government-funded sites such as the U.S. Computer Emergency Response Team at Carnegie-Mellon University, and the manufacturers themselves. Sometimes the problem and the ways to exploit it are reported before a repair patch can be developed. It has become absolutely vital, therefore, that every organization should monitor these information sources to be certain that all relevant holes in security are understood and closed as soon as possible.

If the hole is not closed, and experience indicates that this is often the case, an organization can continue to operate with known holes in its security. It is also vital that the internal and systems auditors understand the importance of monitoring and closing software holes and that this is included in their review plans.

It is important to note that not all updates recommended by vendors are critical security issues. Some are for efficiency, some are to provide new features, while some correct functionality problems. Each organization must evaluate which updates are critical. There is a substantial cost involved in constantly updating systems, and each update could introduce new problems.

6.2 COMPUTER ACCESS CONTROL

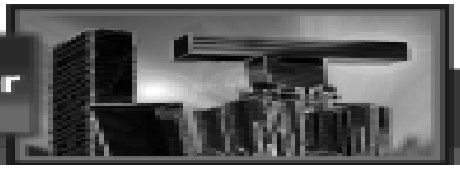
These are controls that allow only authorized persons to gain access to sensitive systems. These include:

6.2.1 Passwords. Use passwords that are long enough to be difficult to guess. Passwords should not be comprised of simple words, names of relatives, and so on, and should be changed regularly. Some organizations have had good results by requiring every password to combine upper and lower case letters, numbers, and special characters.

6.2.2 Compartmentalization. Restrict users to the specific files and programs that they have a job-related need to access. This requires updates as necessary to relate access to needs of people moving from assignment to assignment within the organization.

6.2.3 Use of one-time passwords. Use hardware or software that generates a new password for each access. This may be generated through the passage of time or using a calculator-like device to enter a randomly generated "challenge" number and get a response number that is then entered into the computer to validate identity.

6.2.4 Automatic log off. Use this measure to prevent unauthorized access to the system when authorized users fail to log off.



6.2.5 Use of biometrics. Use fingerprints, iris recognition, hand geometry, and other new technologies for added measures of control.

6.2.6 Time-day controls. Restrict personnel access to those times when they are supposed to be on duty. An extension of this concept for organizations using automated time-clock systems is to deny access and report a violation if access is attempted when an employee is not shown in the time clock system as being present.

6.2.7 Random personal information checks. Implement this means of identifying unauthorized log-in attempts. The system randomly transmits a question that only the authorized individual could answer and denies access unless the right answer is received. If several dozen personal questions are on file, this technique can be very effective.

6.2.8 Internet authentication. Use this control for telecommuting employees. With telecommuting on the rise, many organizations are taking advantage of low-cost, high-bandwidth Internet connections which offer high download speeds for nominal monthly charges, and include continuous access, twenty-four hours a day, seven days a week. This technology, which identifies a specific Internet user and sends information across the Internet securely is rapidly evolving as is cable-television-based broadband Internet access.

6.2.9 Dial-back systems. Use these systems when access is through a dial-up system. On accepting a user ID and password, the system hangs up and dials an established number at which the approved user is standing by. This is very helpful when a person works at a predictable location, for example, the home office of a telecommuting employee.

7.0 PREVENTION IS BETTER THAN CURE

When investigating computer crimes, investigators, systems auditors and forensic accountants often discover what could have been done to prevent the crimes.

It is often found that *the organizations failed to*:

7.1 Have written policies and security rules for the use of computers and systems.

7.2 Have temporary employees and independent contractors follow the same security rules as regular employees.

7.3 Adjust access as people changed responsibilities internally.

7.4 Keep up with and close security holes in application software, firewalls, and operating systems.

7.5 Maintain virus protection on a fully updated basis.

7.6 Have effective policies for security over proprietary information

7.7 Have good interaction between the human relations, systems, and security functions

7.8 Maintain an effective internal accounting controls

7.9 Maintain effective supervision over those with sensitive access to systems

7.10 Have employee instructions on security issues

7.11 Utilize a computer audit software

7.12 Maintain an effective physical security system.

7.13 Have and periodically review a Computer Security Checklist.

8.0 COMPUTER SECURITY CHECKLIST

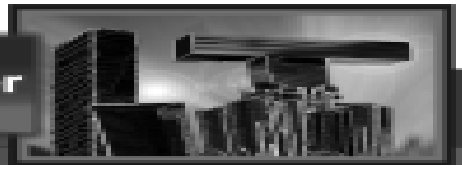
It is of utmost importance for a Systems Manager, Systems or Internal Auditor to develop a checklist which should be reviewed periodically. The essence of this is to be aware on time when circumstances have changed from the previous control level and thus, controls may have become inadequate for the present circumstance.

Some examples of items for such a checklist are provided below. The usual optional answers are: Yes, No, and Not Applicable.

8.1. PHYSICAL SECURITY

a. Computer Room: Most medium and larger organizations will have dedicated computer rooms for their mainframe computers or file servers. The following are basic questions to answer when looking at such a facility:

- Are adequate fire detection and suppression



systems in place? For example, does the computer-room construction have a minimum one-hour fire-resistance rating in addition to smoke detectors, fire alarms, fire extinguishers, and sprinklers?

- If power supplies are unreliable or the nature of processing is critical, are suitable precautions in place? For example, are there battery-equipped Uninterruptible Power Supplies (UPS) or backup generators and surge protectors for PCs?

- Is access to especially sensitive computer installations appropriately restricted? For example, are there key locks, combination or cipher locks, or card-access systems along with access-control policies and procedures?

b. PCs and Workstations

- Are PCs and workstations in areas where theft is a threat secured by cables, locks, or other antitheft devices?

- Are computer users prohibited by written policy from installing unauthorized software including encryption software on company-provided PCs and notebook computers?

8.2 COMMUNICATIONS SECURITY

a. Is a user ID and password system or stronger system with one-time passwords or biometric security in place?

b. Are the ID and password systems properly administered? For example, is the distribution of new IDs controlled and are terminated users promptly deleted from the system?

c. Are users aware of the responsibilities associated with their password? For example, are they required to sign computer-use and confidentiality agreements and are they instructed to maintain password secrecy and not to choose simplistic or easily-guessed passwords?

d. Are passwords changed regularly (for example, every 90 days)?

e. Does the system monitor and control use? For example, by restricting users to specific terminals or specific times, automatically logging out inactive users, limiting the number of log-on attempts, and recording all usage for later follow-up and investigation if required?

8.3. DATA SECURITY

a. Is access to on-line data limited to authorized individuals through built-in software restrictions or screening?

b. For extremely sensitive data, has data encryption been considered as a security measure?

c. Are data files including all backups stored on magnetic media and kept in a physically secure location away from the computers to which only authorized persons are allowed access?

8.4. SOFTWARE INTEGRITY

a. Is access to production versions of all software tightly controlled by a production librarian or similar authorized person?

b. Is access to all software programming code controlled so that programmers or others cannot subsequently alter tested and approved software?

c. Are appropriate policies in place to guard against computer viruses?

8. 5. OPERATIONS SECURITY

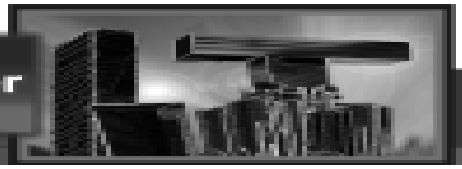
a. Do detailed operator instructions or manuals exist?

b. Is computer activity logged and is any unusual operator activity investigated?

c. Are computer-operations personnel prohibited from altering the program code and the Job Control Language, which match the program and data files to be run?

8.6 SYSTEM RECOVERY

a. Are copies of all data files and software made on a regular basis? For example, are copies made weekly, with daily backups of transaction files?



b. Is at least one backup copy of all data files and software stored off site?

c. In the case of catastrophic failure, do alternative processing arrangements exist?

The above are just examples of the types of questions that could be contained in a comprehensive Computer Security Checklist for an organization.

9.0 CONCLUSION

In this paper we have discussed the controls for preventing and detecting computer crime. As responsible ICT practitioners, our goal should be to ensure that our various organizations suffer as close as possible to zero level of loss in monetary value and in terms of physical assets. To achieve this, we need to focus more on preventive measures. As the saying goes, "prevention is better than cure." According to the Association of Certified Fraud Examiners' (ACFE) 2004 Report to the Nation on Occupational Fraud and Abuse (p.iv) "the most cost-effective way to deal with fraud is to prevent it. According to our study, once an organization has been defrauded, it is unlikely to recover its losses. The median recovery among victim organizations in our study was only 20% of the original loss. Almost 40% of victims recovered nothing at all."

There is no doubt that one of the major challenges that will continue to confront the ICT professional for the foreseeable future will be in the area of computer fraud and abuse. Therefore, let us all rise up to this challenge by adequately equipping ourselves with the necessary tools and techniques for combating this menace. Then let us ensure that necessary measures as discussed above are implemented at our various organizations.

REFERENCES

AVEY, TEDD "Computer Crime and Computer Criminals." The CPA's Handbook of Fraud and Commercial Crime Prevention, New York: The American Institute of Certified Public Accountants, Inc., 2002.

OLAYIWOLA, PETER O. "*Accounting Information, Global Fraud & Allied Issues."* A Paper presented at the Mandatory Continuous Professional Education of the Institute Of Certified Public Accountants Of Nigeria (ICPAN) held at the International Conference Centre, Etiebets' Place (2nd Floor), Ikeja, Lagos, Nigeria, 1st - 2nd December, 2004

OLAYIWOLA, PETER O. "Electronic Payment Systems: A New Area of Challenge for Professional Accountants." A Paper presented at the Induction Ceremony for New Members of the Institute of Certified Public Accountants of Nigeria (ICPAN) held at Airport Hotels, Ikeja on Friday, 21st November, 2003.

OLAYIWOLA, PETER O. "The Computer Audit Function." A Paper presented at the Conference Organized for Heads of Internal Audit Units in the State Civil/ Public Service, Local Government Council, Tertiary Institutions and Staff of Inspectorate and Compliances (State Treasury Office) at Badagry, 9 - 11 October, 2002.

OLAYIWOLA, PETER O. "Establishment and Managing the EDP Audit Function: Public Sector Perspective." A Paper presented at A- 3 Day Workshop on Effective Financial administration in the Public Sector, held at Hamdala Hotel, Kaduna, 24th - 26th June, 1998

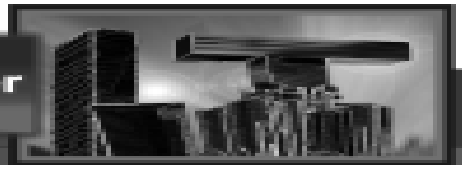
OLAYIWOLA, PETER O. "Banking in Nigeria: Fraud Detection, Impact of Information Technology and Internal Control." A Paper Presented at a Seminal organized by Obafemi Awolowo University Consultancy Services, held at Federal Palace Hotel, Lagos, 15th December, 1998

OLAYIWOLA, PETER O. "The Role of Internal Control in the Detection and Prevention of Fraud." A Paper presented at MIT97 the 6th International Conference and AGM of the Computer Association of Nigeria (COAN) held at Premier Hotel, Ibadan, May 28th - 30th, 1997

THE ASSOCIATION OF CERTIFIED FRAUD EXAMINERS. "Choicepoints Error Sparks Talk on ID Theft Law". FraudInfo, February 24, 2005.

THE ASSOCIATION OF CERTIFIED FRAUD EXAMINERS. "ACFE European's Council on Occupational Fraud and Abuse Takes First Steps". FraudInfo, March 10, 2005

THE ASSOCIATION OF CERTIFIED FRAUD EXAMINERS. 2004 Report to the Nation on Occupational Fraud and Abuse. Austin, Texas, 2005



Capacity Planning for E-banking Systems

John B. Oladosu; Justice O. Emuoyibofarhe; A. Akinkunmi

ABSTRACT

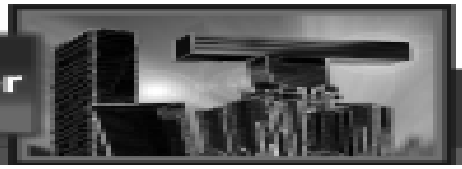
E-banking applications are fast evolving. It is one of the greatest competitive edges in the banking industry today. As more clients make use of the location and time unconstrained advantage of E-banking systems, greater demands are imposed on application, network and other infrastructural capacity requirements. Undesirable system performance resulting from capacity limitations could result in very high users complaints. E-banking system capacity may become inadequate over a short period. These inform the necessity for implementers of E-banking systems to take a proactive evaluation of the demand on E-banking systems capacity in order to forestall undesirable systems performance occasioned by limited infrastructural capacity. Capacity planning is the way out of this problem. Capacity planning is a performance-tuning tool applicable in such situation as explained above. In this work, we take a general study of Capacity Planning as it applies to E-banking application systems. We developed a simulation to determine the maximum number of users an E-banking system can tolerate under specific tolerance rules for an average user. Our system could also be used to simulate the required system capacities for varying hypothetical user bases. The results are used to predict potential point of failure and to recommend capacity upgrades. Also, future capacity needs could be analysed based on hypothetical changes intended or growth expected. The results of our simulation is used to advise banking managements estimate capacity requirement for a specific level of performance under an estimated expected network load. A futuristic approach should be applied by a bank wanting to use simulations of our system in order to get best results.

Introduction

The success or otherwise of any software lies in the hardware facilities supporting such software. E-banking applications are no exceptions. The demand on Banks to remain in business is more pressing than ever before. E-banking is one of the tools that would assist any bank to achieve this goal. This makes the development of such application critical for any Bank. However, it is worth noting that optimum and continued functionality of software is as important as its development. This makes capacity planning for software a critical issue at this stage of technological development.

Capacity planning involves balancing capacity with demand. Capacity is the amount of traffic or processing that the available infrastructure is capable of supporting; while demand is the load that users or applications place upon that infrastructure.[3]. Since E-banking is a network based software, it is pertinent to obtaining accurate information about the network in order to have successful capacity planning. Capacity planning is used to evaluate the current trends in resource usage and plan for growth and changes *before* problems occur.[3] Capacity planning can serve as a reality check, prompting an organization, a bank in this case, to come to terms with its current business plan and its goals for the future.

Four ways of predict e-commerce system performance are identified in [2] as follows: Statistical forecasting, User simulation, Profile and Synthetic Workload. Profiling method is used in this work. Our work is an attempt at giving an input to development of enduring E-banking applications in the achievement of the goals aforementioned. We start by examining theoretical background to the work on Capacity Planning. A brief statement of our rationale for the research follows this. Next, we present the report of simulations we carried out using Benchmark Factory 4.5 together with the analysis of results. Finally, we present our conclusion and future direction in our work.



Performance Statistics and Metrics

In this section, we shall consider performance statistics and metrics relevant to our discussion as presented in [1]. There are four main reasons for large errors in estimations of performance metrics for cluster servers:

- There are few or no operational statistics from a production or test system available,
- The baseline statistics were collected on a legacy system with a different architecture and there are no comparative benchmarks or workload independent capacity measures,
- The contribution of different components of the workload to the overall system performance is not sufficiently described,
- The impact of remote, global cache memory references cannot be estimated for lack of traces and understanding of their impact.

The article further identified the following system or component level statistics and derived metrics, which can be used in the estimation of capacity.

CPU Consumption

Service Time

I/O Requirements

Memory Consumption

Network Throughput and Latency

Data Referencing Behaviour

CPU Consumption

Consisting of the following metrics:

- The total CPU power used at a steady state per unit of time

  Given as

  Total CPU consumed/sec = #CPUS * CPU clock rate * pct utilization —— (1)

- The unit-of-work CPU consumption per unit of time (e.g., CPU sec/transaction, CPU cycles/transaction etc.)

  i.e: (Total CPU consumed/sec) / (unit of work/sec) ——————————— (2)

  Where #CPU = the number of CPUs, pct utilization = the peak utilization of the system

  The graphical results of our simulations can be used to estimate the aforementioned metrics.

Service Time per Transaction

Consisting of the following:

The database service time is the total time spent executing in the database, processing or waiting for a resource, and thus is given by

- Service Time = CPU Time + Wait Time ———————————— (3)

  The total estimated service time per business transaction is

- (Total wait time/sec) / (Transactions/sec) ———————— (4)

  Summarily, based on the database statistics, the estimated times for a business transaction are as follows:

  DB Service Time DB CPU Time Total CPU Time DB IO wait time

I/O Requirements

Consisting of the following metrics

- I/O rates in operations per sec
- I/O throughput in MB/sec
- I/O sizes for reads and writes
- Average disk service times
- Redo write rate and size

The redo write rate and size of the logs are used to derive the volume of redo that would have to be archived in the event that archiving to a recovery area was required.

Memory Consumption

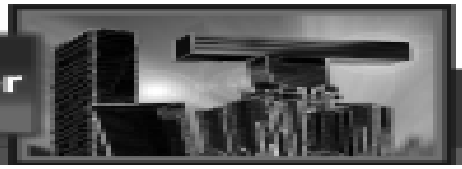
This statistic is the total buffer cache and process memory

Network Throughput and Latency

The fact that multiple clusters or database are sharing the same physical interconnect should be taken into account. However, it is recommended to configure different physical interconnects or VLANs for each cluster managed in the server clusters.

Data Referencing Behaviour

Data referencing patterns and distributions can be important issue for globally shared data. While a single system with large memory support may cache the entire



working set of an application, the same cache may have to be distributed over multiple machines in a commodity cluster and data may have less locality because it is shared by multiple machines and must therefore be globally synchronized.

CPU Capacity Estimates For Interconnect Traffic

This is particularly relevant in a grid computing environment where some memory references may be non-local due to contention or a large shared working set and database blocks are read through the private interconnect via memory-to-memory copies.

Estimating the CPU Impact of Global Cache Coherence

The impact of the global cache processing is not known in advance. Therefore, in [1] the cost is estimated the by

- Adjusting the CPU per transaction by an added, assumed percentage increase and then
- Computing the transaction rate with the adjusted metric for different system utilizations.

Then the number of nodes required to support the target transaction rate is calculated by

- Dividing the target transaction rate by the adjusted transaction rate.

According to [1], the assumption that the clock rate is the only quantifier to distinguish one CPU from another could result in a large error. In reality fewer servers of each kind are needed to sustain the throughput and the overcapacity resulting from the initial estimate could be used to scale up the workload and achieve higher throughput.

Methodology for Determining Bandwidth Requirements for Network Components of an E-Banking Server

Here we present methodology for determining bandwidth requirements for network components of a World Wide Web server as provided in [10] which by extension can be applied to E-banking server owing to the similar nature of their dynamic content and impact of user/network characteristics.

The following bandwidths are considered with the respective denotation:

$\mathbf{B_{NTW}}$ representing network bandwidth

$\mathbf{B_{SND}}$ representing send bandwidth

$\mathbf{B_{RCV}}$ representing receive bandwidth

We can obtain by simply adding the required bandwidths over proxy to client and proxy to remote server connections. Noting that a client request may occasionally require forwarding the request to a remote server and obtaining the response before data is sent to the client, we have:

$$B_{NTW} = B_{SND} + B_{RCV} \quad \ldots\ldots(5)$$

Where

$$B_{SND} = [L_{res} + q_{rrf}(L_{req} + A_{rcv}) + (M_{snd} + q_{rrf} M_{rcv})\theta_{ka}](1 + n_{rex}) \quad \ldots\ldots(6)$$

and

In which,

L_{req} is the average size of a client request in bytes.

L_{res} is the average size of a server response in byte.

$L_{res} = L_{snd}N_{snd}$ where L_{snd} is the average send packet size (including headers) and N_{snd} is the average number of send packet per file.

A_{rcv} is the network receives explicit acknowledgement overhead per transaction.

M_{rcv} is the network receives management overhead per transaction.

$$T_{SND} = [N_{snd} + N_{rex}(N_{snd} + q_{rrb}N_{sendack}) + (2(q_{rrf} + q4)\theta_{ka}](1 + n_{rex}) \quad \ldots\ldots(8)$$

M_{snd} is the network sends management overhead per transaction.

q_{rrb} is the remote receive fraction in byte i.e. fraction of response data per request, in byte.

q_{rrf} is the remote receive fraction in file counts

n_{rex} is the average number of retransmission that a message may suffer due to transmission errors or large delays in acknowledgements.

θ_{ka} is the probability of not keeping the connection alive.

If we represent bandwidth in transmission with symbol T instead of B then, bandwidth in transmission units or packets are obtained from (6) and (7) as follows:

and

$$T_{NTW} = T_{SND} + T_{RCV} \quad \ldots\ldots(10)$$

Rationale for the Work

Several work has been done on capacity planning [1],[2],[3],[8],[9],[10],[14]. However, none, to the best o our knowledge, addressed capacity planning with specific emphasis on E-banking systems especially as simulating their (E-banking) real world behaviour such as we have done this work. We also have our immediate locality (the Nigerian banking industry) in mind in our research. This makes our proposed methods more relevant to our immediate system.

Simulation Configurations of Users/Workload scenarios

Our simulation set-up was configured to run under a maximum of 20000 virtual user scenario. The Profile consisted of an SQL Server 2000 driven experimental E-banking application. The following Scalable hardware jobs were performed on the profile with the maximum tolerable virtual users:

- Scalable CPU Intensive Test
- Scalable Hardware Read Intensive Test
- Scalable Hardware Update Intensive Test
- Scalable Hardware Insert Intensive Test
- Scalable Mixed Workload (CPU, Update, Read, Insert Tests)

Analysis of Results

- Scalable CPU Intensive Test Graphical Results

The following results were obtained under this test. Fig. 1 indicates that for the CPU under test for the maximum of 20 concurrent users, throughput dropped between 12 and 16 users. This indicates a potential point of failure after 12 users. Assuming 60ms degree of tolerance for an average E-banking system user; we infer from fig. 2 that the system performs best under 13 concurrent users.

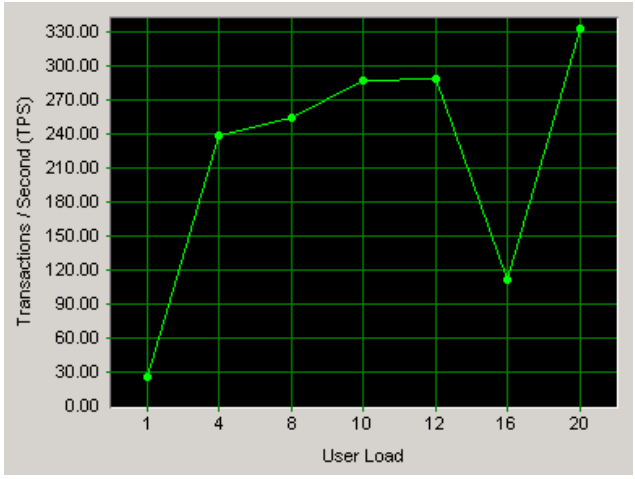


Fig. 1: CPU Intensive Test Transactions Per Second (TPS)/ User Load

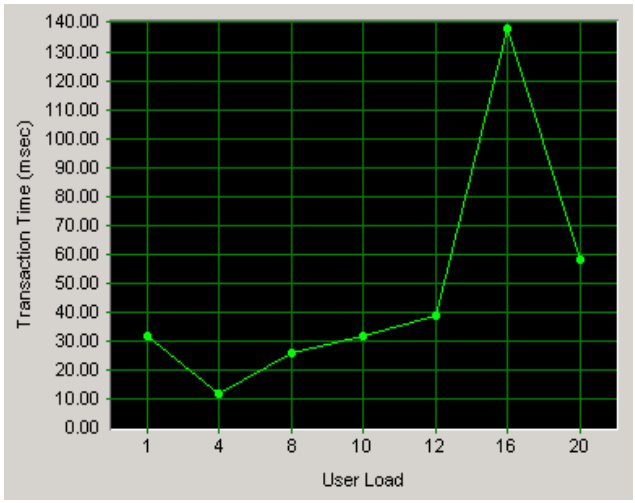


Fig. 2: CPU Intensive Test Transactions Time/User Load

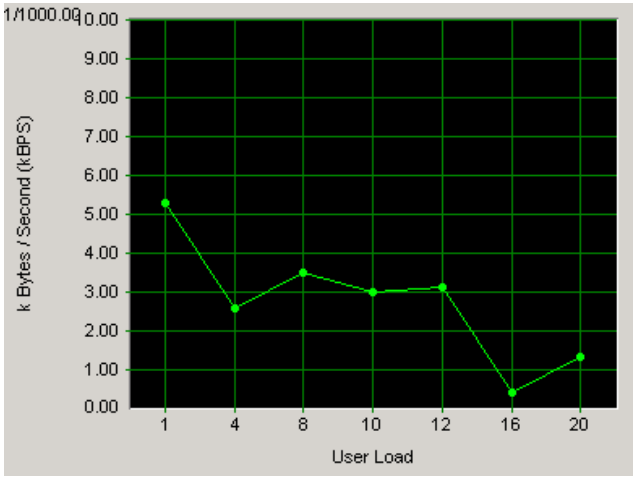


Fig. 3: CPU Intensive Test Kilobyte per second (KBPS)/ User Load

Fig. 3 shows further that the system bandwidth drops considerably after 12 concurrent users.

We can see a good level of consistency in the results analysed above.

- **Scalable Read Intensive Test Graphical Results**

Figs. 4 and 5 indicate that the system under test is read efficient for the maximum 20 users accessing the E-banking system. Hence no potential point of failure for read intensive operation under this hardware configuration.

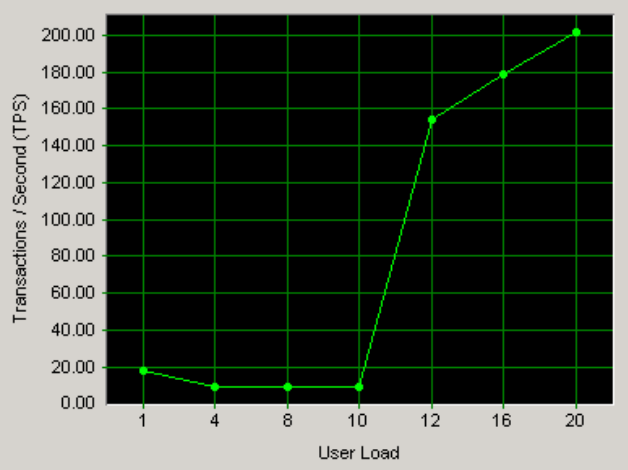


Fig. 4: Read Intensive Test Transactions Per Second (TPS) / User Load

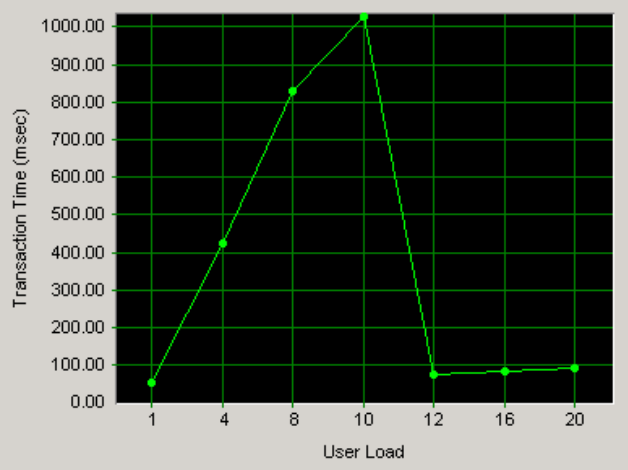


Fig. 5: Read Intensive Test Transactions Time / User Load

- **Scalable Update Intensive Test Graphical Results**

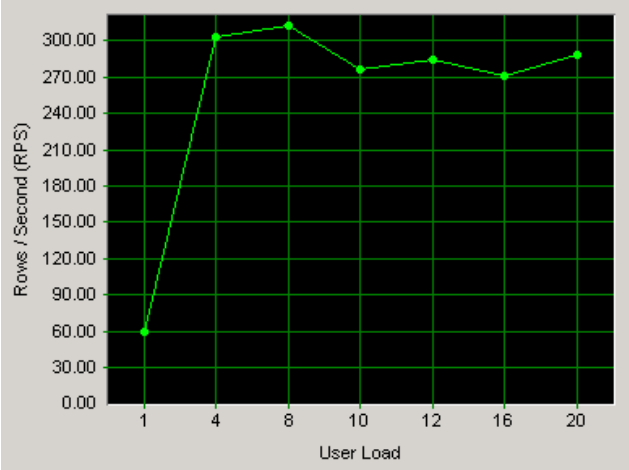


Fig. 6: Update Intensive Test Rows Per Second (RPS) / User Load

Fig. 6 informs us that our system becomes update inefficient after 8 concurrent users. More rows will be generated as number of concurrent users increase. The fact that no appreciable increase in rows processed per second is achieved after 8 concurrent users gives an indication of potential point o failure. Assuming again a maximum of 60ms level of tolerance for an average user, the system will reach the peak of its capacity after 17 concurrent users as can be seen from fig. 7.

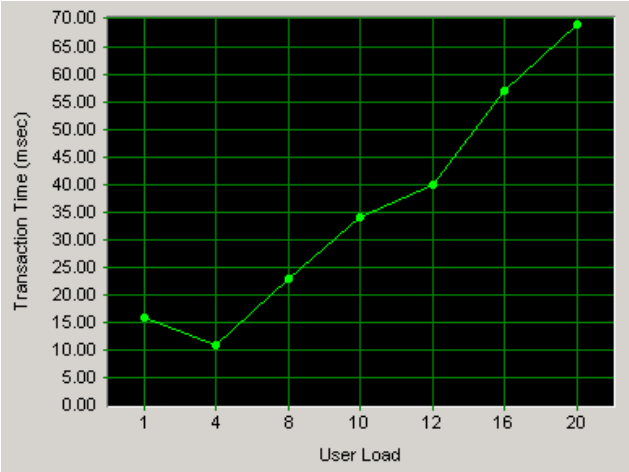


Fig. 7: Update Intensive Transaction Time / User Load

- **Scalable Insert Intensive Test Graphical Results**

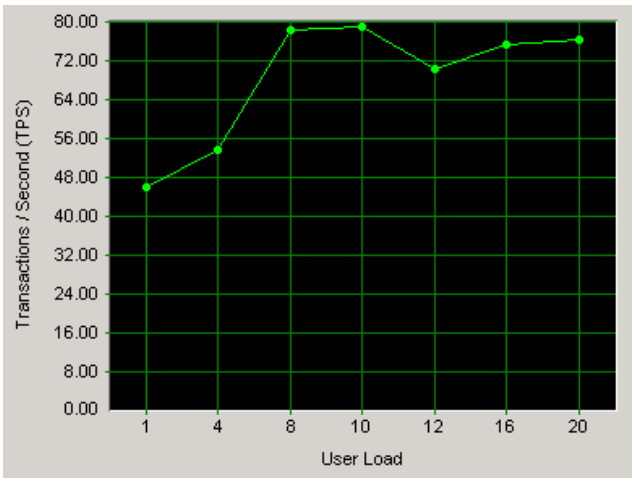


Fig. 8: Insert Intensive Test Transactions Per Second (TPS) / User Load

Again we see from fig. 8 that the system under test becomes inefficient after 8 concurrent users performing insert operation.

- **Scalable Mixed Workload (CPU, Update, Read, Insert Tests) Graphical Results**

Fig. 9 shows that for mixed workload, the system under test is at its best for 1 to 9 users assuming our normal 60ms tolerant level.

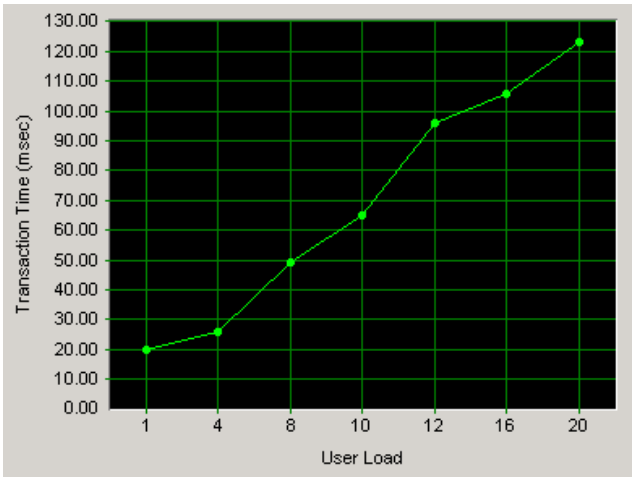


Fig. 9: Mixed Workload Transaction Time / User Load

Conclusion and Further Direction

The methods presented in this paper can be used to test any configuration of hardware yielding performance expectations of the E-banking system. Potential point of failures can be detected before they really occur thus enabling implementers to recommend best configurations for the system under design before actual rollout. For system already implemented, needed capacity upgrade can be estimated to a very high degree of accuracy. In a further work in this research, we intend to use case studies from banks that are already implementing E-banking applications in Nigeria. Real life data will be used to perform simulations; which results will advise if the banks' current capacities will stand the test of time on expected usage growth. If not, recommendations will be made on capacity upgrades.

References

1. Sftean Roesch, Michael Zoll, Srinivasan Subramaniam and Akira Hangai, "*Project MEGAGRID: Capacity Planning For Large Commodity Clusters"* An Oracle, Dell, EMC, Intel Joint White Paper December 2004
2. Dennis Sarris and Jim Hofer, "*Capacity Planning for e-Commerce Systems With Benchmark Factory™*" Benchmark Factory White Paper 2004
3. *e*Health Release 5.0; 2004
4. Capacity planning ensures success of new service http://www.intel.com/internetservices/intelsolutionservices
5. Worldwide business solutions www.geac.com 2002
6. John Holmes, "*Capacity Planning made Easier"* First Data Resources Ltd 2002
7. Ludmila Cherkasova, Wenting Tang, and Sharad Singhal, "*An SLA-Oriented Capacity Planning Tool for Streaming Media Services,"* Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN'04) © 2004 IEEE
8. Andras Farago, "*Network Level Capacity Planning with Efficiently Computable Global Optimum,*" Proceedings of the 10th IEEE Int.l Symp. on Modeling, Analysis, & Simulation of Computer & Telecommunications Systems (MASCOTS.02) © 2002 IEEE.
9. L. Vekiarides, D. Finkel, "*NETCAP: A Tool for the Capacity Planning of Ethernet LANs,*" Sixth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, July 1998, pp. 198.
10. Krishna Kant, Youjip Won, "*Server Capacity Planning for Web Traffic Workload,"* IEEE Transactions on Knowledge and Data Engineering, September 1999, pp. 731-747.
11. Robert Lackman, ShengJun Li, "*Finite Population Task Oriented Queueing: A Model for Internet Server Capacity Planning*," 2001 International Conference on Computer Networks and Mobile Computing (ICCNMC'01), October 2001, pp. 217.
12. Bin Zhou, Hussein T. Mouftah, "*Spare Capacity Planning Using Survivable Alternate Routing for Long-Haul WDM Networks*," Seventh International Symposium on Computers and Communications (ISCC'02), July 2002, pp. 732.
13. Virgílio A.F. Almeida, Daniel A. Menascé "*Capacity Planning: An Essential Tool for Managing Web Services,"* IT Professional July 2002 pp. 33-38.
14. Neil J. Gunther "*Hit-and-Run Tactics Enable Guerrilla Capacity Planning,"* IT Professional July 2002 pp. 40-46.

CGI Programming on the World Wide Web

Adesesan Barnabas Adeyemo, Oladipo, Onaolapo Francisca

ABSTRACT

The Common Gateway Interface, CGI is the part of the Web server that can communicate with other programs running on the server. It emerged as the first way to present dynamically generated information on the World Wide Web thereby allowing the computer to generate Web pages instantly at the user's request rather than being written by someone in advance. With CGI, one can display sales figures for particular products month by month, as requested by a staff, using beautiful pie charts or plots, customers can enter keywords in order to find information on products and one can also offer day-to-day conveniences, like collecting comments from users, offering them searches through archives and so on. CGI is trivial in design, and anyone with an iota of programming experience can write rudimentary scripts that work. This paper offers a comprehensive explanation of CGI and related techniques for people who want to provide their own information servers on the Web. It starts at the beginning, explaining the value of CGI and how it works, then moves swiftly into the subtle details of programming by looking at Input/ Output, Forms, Server Side Includes, Hypermedia Documents, Multiple Form Interaction, Gateways, Databases, and Search/Index Utilities. We conclude by examining Debugging and Testing of CGI Applications.

1.1 Introduction

Common Gateway Interface, CGI is the part of the Web server that can communicate with other programs running on the server[5]. With CGI, the Web server can call up a program, while passing user-specific data to the program (such as what host the user is connecting from, or input the user has supplied using HTML form syntax). The program then processes that data and the server passes the program's response back to the Web browser. CGI is not magic; it is just programming with some special types of input and a few strict rules on program output. People at many different levels of computer expertise come to CGI to solve their problems. CGI documents could consist of, among other things, forms that ask for feedback or registration information, image maps that allow the user to click on various parts of the image, counters that display the number of users that accessed the document, and utilities that allow users to search databases for particular information.

CGI programming can be carried out using C, C++, TCL etc [3], but the language of choice for CGI programming in this paper is the Practical Extraction and Reporting Language. (Perl), at least on UNIX systems. Since everyone is not expected to be a Perl programmer, the descriptions are given in plain English, so that the techniques can be implemented in any language of choice. The main thing to learn is the concept and implementation. There is a preference for the UNIX operating system environment in this paper. This is simply because UNIX is the most popular system for Web servers [3]. Some of the things that are done all the time on UNIX-such as pipe output to another program-have to be done differently on other systems. Similarly, some of the security concerns that go along with executing UNIX commands would not exist on other systems. However the basic CGI tasks are the same in any language, on any system [2]. They spring from the division of labor between client and server, and the protocols they use to communicate. CGI is trivial in design, and anyone with a little programming experience can write rudimentary scripts that work [3]. It is only when the needs are more demanding that the programmer have to master the more complex workings of the Web. Of course, there are special techniques that are particular to CGI, and that is what this paper is mostly about. Figure 1 shows a simple model of CGI.

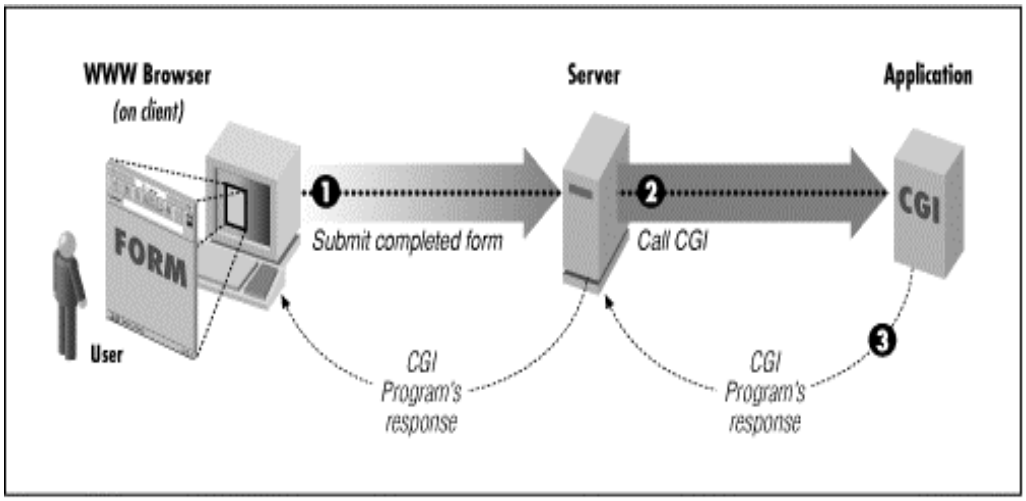


Figure 1: a simple CGI model

1.2 CGI Applications

CGI turns the Web from a simple collection of static hypermedia documents into a whole new interactive medium, in which users can ask questions and run applications [3]. The web pages are transformed from primarily text and graphics that the readers passively browse to interactive presentations that can provide different options based on readers' input. The following are some of the possible applications that can be designed using CGI.

1.2.1 Forms

One of the most prominent uses of CGI is in processing forms [3]. Forms are a subset of HTML that allow the user to supply information. In a document-based query, the input to the script is in form of whatever the reader types into the search window [2]. If there are spaces in the input, those spaces create separate arguments in the call to the gateway script. In form-based queries, things are slightly different, because the user can retrieve a much greater variety of information from a form. The forms interface makes Web browsing an interactive process for the user and the provider. Figure 2 shows a simple form.

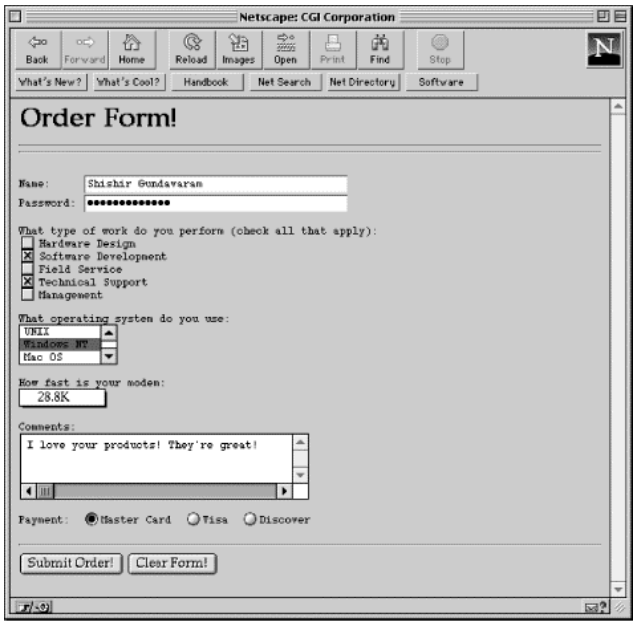


Figure 2: Simple form illustrating different widgets

As can be seen from the figure, a number of graphical widgets are available for form creation, such as radio buttons, text fields, checkboxes, and selection lists. When the form is completed by the user, the Submit Order! button is used to send the information to the server, which executes the program associated with the particular form to "decode" the data. Generally, forms are used for two main purposes. At their simplest, forms can be used to collect information from the user. But they can also be used in a more complex manner to provide back-and-forth interaction. For example, the user can be presented with a form listing the various documents available on the server, as well as an option to search for particular information within these documents. A CGI program can process this information and return document(s) that match the user's selection criteria.

1.2.2 Gateways

A gateway script, mostly is a program that is run on a Web server, triggered by input from a browser [6]. The gateway script is usually a link between the server and some other program running on the system; for example a database. Web gateways are programs or scripts used to access information that is not directly readable by the client. For example, if you have an

Oracle database that contains baseball statistics for all the players on your company team and you would like to provide this information on the Web. It is evident that you certainly cannot point your client to the database file (i.e., open the URL associated with the file) and expect to see any meaningful data. CGI provides a solution to the problem in the form of a gateway. You can use a language such as *oraperl* or a DBI extension to Perl to form SQL queries to read the information contained within the database. Once you have the information, you can format and send it to the client. In this case, the CGI program serves as a gateway to the Oracle database, as shown in Figure 3.

Similarly, you can write gateway programs to any other Internet information service, including Archie, WAIS, and NNTP (Usenet News). In addition, you can amplify the power of gateways by using the forms interface to request a query or search string from the user to retrieve and display *dynamic*, or *virtual,* information.

1.2.3 Virtual Documents

Virtual, or dynamic, document creation is at the heart of CGI 5]. Virtual documents are created on the fly in response to a user's information request. The programmer can create virtual HTML, plain text, image, and even audio documents. The following is a simple example of a virtual document:

> Welcome to NCS's WWW Server!
> You are visiting from demo.com. The load average on this machine is 1.25.
> Happy navigating!

In this example, there are two pieces of dynamic information: the alphanumeric address (IP name) of the remote user and the load average on the serving machine. This is a very simple example. On the other hand, very complex virtual documents can be created by writing programs that use a combination of graphics libraries, gateways, and forms.

2.0 Internal Workings of CGI

Most servers expect CGI programs and scripts to reside in a special directory, usually called *cgi-bin*, and/or to have a certain file extension [3]. When a user opens a URL associated with a CGI program, the client sends a request to the server asking for the file. For the most part, the request for a CGI program looks the same as it does for all Web documents. The difference is that when a server recognizes that the address being requested is a CGI program, the server does not return the file contents verbatim. Instead, the server tries to

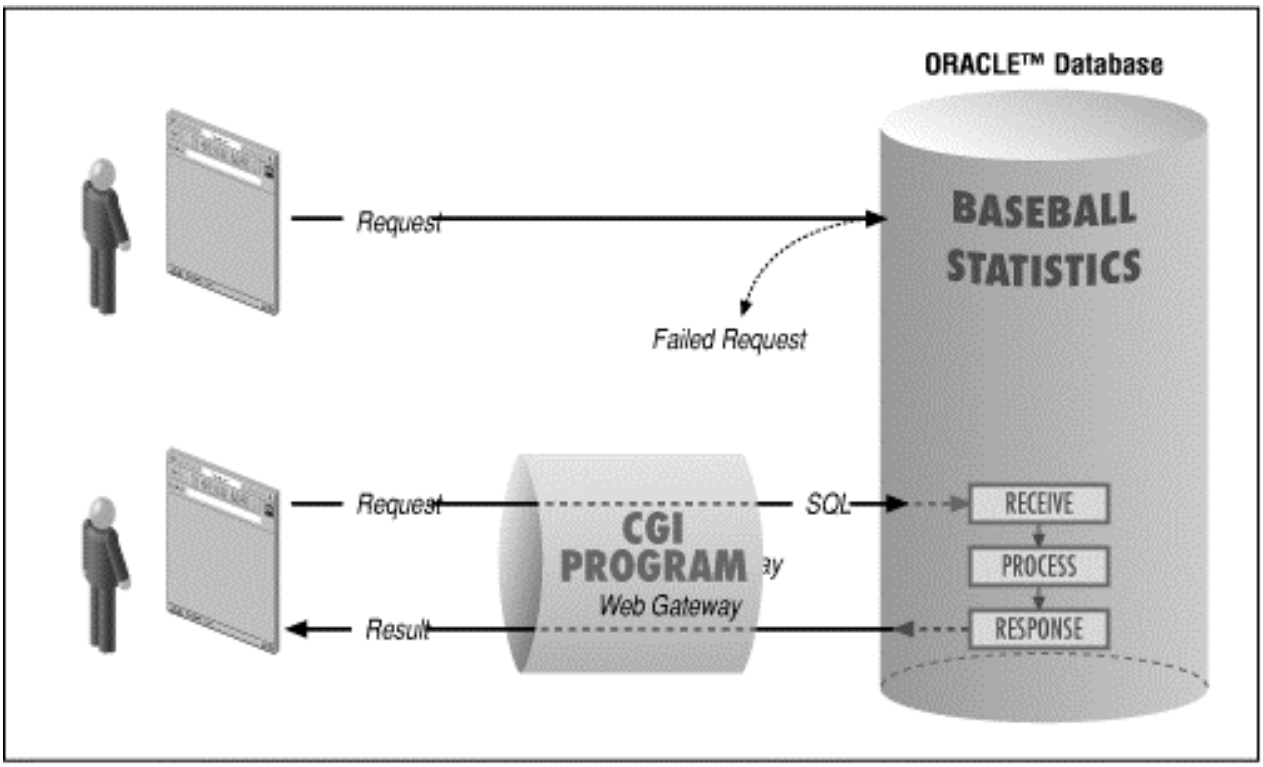


Figure 3: A gateway to a database

execute the program. Here is what a sample *client* request might look like:

```
GET /cgi-bin/welcome.pl HTTP/1.0
Accept: www/source
Accept: text/html
Accept: image/gif
User-Agent: Lynx/2.4 libwww/2.14
From: demo@ui.edu
```

This GET request identifies the file to retrieve as */cgi-bin/welcome.pl*. Since the server is configured to recognize all files in the *cgi-bin* directory tree as CGI programs, it understands that it should execute the program instead of relaying it directly to the browser. The string *HTTP/1.0* identifies the communication protocol to use. The client request also passes the data formats it can accept (*www/source*, *text/html*, and *image/gif*), identifies itself as a Lynx client, and sends user information. All this information is made available to the CGI program, along with additional information from the server.

The way CGI programs get their input depends on the server and on the native operating system [3]. On a UNIX system, CGI programs get their input from standard input (STDIN) and from UNIX environment variables [3]. These variables store such information as the input search string (in the case of a form), the format of the input, the length of the input (in bytes), the remote host and user passing the input, and other client information. They also store the server name, the communication protocol, and the name of the software running the server. Once the CGI program starts running, it can either create and output a new document, or provide the URL to an existing one. On UNIX, programs send their output to standard output (STDOUT) as a data stream. The data stream consists of two parts. The first part is either a full or partial HTTP header that (at minimum) describes what format the returned data is in (e.g., HTML, plain text, GIF, etc.). A blank line signifies the end of the header section. The second part is the body, which contains the data conforming to the format type reflected in the header. The body is not modified or interpreted by the server in any way.

A CGI program can choose to send the newly created data directly to the client or to send it indirectly through the server. If the output consists of a complete HTTP header, the data is sent directly to the client without server modification..) Or, as is usually the case, the output is sent to the server as a data stream. The server is then responsible for adding the complete header information and using the HTTP protocol to transfer the data to the client. The following program is a sample output generating an HTML virtual document, with the complete HTTP header:

```
HTTP/1.0 200 OK
Date: Thursday, 22-February-96 08:28:00 GMT
Server: NCS/1.4.2
MIME-version: 1.0
Content-type: text/html
Content-length: 2000
<HTML>
<HEAD><TITLE>Welcome to NCS's WWW
Server!</TITLE></HEAD>
<BODY>
<H1>Welcome!</H1>
.
.
</BODY>
</HTML>
```

The header contains the communication protocol, the date and time of the response, the server name and version, and the revision of the MIME protocol. Most importantly, it also consists of the MIME content type and the number of characters (equivalent to the number of bytes) of the enclosed data, as well as the data itself. Now, the output with the partial HTTP header: MIME (Multipurpose Internet Mail Extensions) is a specification that was originally developed for sending multiple types of data through electronic mail. MIME types are used to identify types of data sent as content over the Web.

```
Content-type: text/html
<HTML>
<HEAD><TITLE>Welcome to NCS's WWW
Server!</TITLE></HEAD>
<BODY>
<H1>Welcome!</H1>
.
.
</BODY>
</HTML>
```

In this instance, the only header line that is output is the *Content-type* header, which describes the MIME format of the output. Since the output is in HTML format, *text/html* is the content type that is declared. Most CGI programmers prefer to supply only a partial header. It is much simpler to output the format and the data than to formulate the complete header information, which can be left to the server. However, there are times when one needs to send the information directly to the client (by outputting a complete HTTP header),

3.0 Configuring the Server

Before a user can run CGI programs on a server, certain parameters in the server configuration files must be

modified. For instance there is a need to first set the *ServerRoot* directive in the *httpd.conf* file to point to the directory where the server software is located:

ServerRoot /usr/local/etc/httpd

4.0 Running CGI Scripts

The *ScriptAlias* directive in the server resource map file (*srm.conf*) indicates the directory where CGI scripts are placed.

ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-bin/

For example, if a user accesses the URL:

http://your_host.com/cgi-bin/welcome

the local program:

/usr/local/etc/httpd/cgi-bin/welcome

will be executed by the server. There can be multiple directories to hold CGI scripts:

ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-bin/

ScriptAlias /my-cgi-bin/ /usr/local/etc/httpd/my-cgi-bin/

The most important reason for placing all CGI programs in distinct directories is system security. By having all the programs in one place, a server administrator can control and monitor all the programs being run on the system. However, there are directives that allow programs to be run outside of these directories, based on the file extension. The following directives, when placed in the *srm.conf* configuration file, allow the server to execute files containing *.pl*, *.sh*, or *.cgi* extensions.

AddType application/x-httpd-cgi .pl .sh .cgi

However, this could be very dangerous. By globally enabling all files ending in certain extensions, there is a risk that novice programmers might write programs that violate system security (e.g., printing the contents of important system files to standard output).

5.0 Programming in CGI

Programming in CGI can involve any language, although certain languages are more suited for CGI programming than others. Before choosing a language, the programmer must consider the following features:

- Ease of text manipulation
- Ability to interface with other software libraries and utilities
- Ability to access environment variables (in UNIX)

Most CGI applications involve manipulating text some way or another, so inherent pattern matching is very important [3]. For example, form information is usually "decoded" by splitting the string on certain delimiters. The ability of a language to interface with other software, such as databases, is also very important. This greatly enhances the power of the Web by allowing the programmer to write gateways to other information sources, such as database engines or graphic manipulation libraries. The last attribute that must be taken into account is the ease with which the language can access environmental variables. These variables constitute the input to the CGI program, and thus are very important. Some of the more popular languages for CGI programming include AppleScript, C/C++, C Shell, Perl, Tcl, and Visual Basic. Here is a quick review of the advantages and, in some cases, disadvantages of some selected languages.

C/C++ (UNIX, Windows)

C and C++ are very popular with programmers, and some use them to do CGI programming. These languages are not recommended for the novice programmer; C and C++ impose strict rules for variable and memory declarations, and type checking. In addition, these languages lack database extensions and inherent pattern-matching abilities, although modules and functions can be written to achieve these functions. However, C and C++ have a major advantage in that you can compile your CGI application to create a binary executable, which takes up fewer system resources than using interpreters (like Perl or Tcl) to run CGI scripts.

C Shell (UNIX Only)

C Shell lacks pattern-matching operators, and so other UNIX utilities, such as *sed* or *awk*, must be used whenever the programmer wants to manipulate string information. However, there is a software tool, called *uncgi* and written in C, that decodes form data and stores the information into shell environment variables, which can be accessed rather easily. Obviously, communicating with a database directly is impossible, unless it is done through a foreign application. C Shell has some serious bugs and limitations that make using it a dangerous proposition for the beginner.

Perl (UNIX, Windows)

Perl is by far the most widely used language for CGI programming. It contains many powerful features, and is very easy for the novice programmer to learn. The advantages of Perl include:

- It is highly portable and readily available.
- It contains extremely powerful string manipulation operators, as well as functions to deal with binary data.
- It contains very simple and concise constructs.
- It makes calling shell commands very easy, and provides some useful equivalents of certain UNIX system functions.
- There are numerous extensions built on top of Perl for specialized functions; for example, there is *oraperl*(or the DBI Extensions), which contains functions for interfacing with the Oracle database.

Perl is used for most of the examples presented in this paper.

Tcl (UNIX Only)

Tcl is gaining popularity as a CGI programming language. Tcl consists of a shell, *tclsh*, which can be used to execute scripts. Like Perl, *tclsh* also contains simple constructs, but is a bit more difficult to learn and use for the novice programmer. Like Perl, Tcl contains extensions to databases and graphic libraries. It also supports regular expressions, but is quite inefficient in handling these expressions at compile time, especially when compared to Perl.

Visual Basic (Windows Only)

Visual Basic is perfect for developing CGI applications because it supports numerous features for accessing data in the Windows environment. This includes OLE, DDE, Sockets, and ODBC. ODBC, or Open Database Connectivity, allows one to access a variety of relational and non-relational databases. The actual implementation of the Windows CGI interface determines how CGI variables are read from a Visual Basic program. However, Visual Basic lacks powerful string manipulation operators.

6.0 CGI Considerations

It is important that the information to be presented is fully understood. If it is plain text or HTML, there is no problem. However, if the data is unreadable by the client, a gateway has to be written to effectively translate that data. This leads to another important matter: The original (or "unreadable") data has to be organized in such a way that it will be easy for the gateway to read from and write to the data source. Once you have the gateway and you can retrieve data, you can present it in numerous ways. For example, if the data is numerical in nature, the programmer can create virtual graphs and plots using various utility software. On the other hand, if the data consists of graphical objects, one can modify the information using numerous graphic manipulation tools. To ensure the creation of effective virtual documents, the programmer needs to think about what to present and how to present it long before the actual process of implementing CGI programs [1].

7.0 Input to the Common Gateway Interface

When a CGI program is called, the information that is made available to it can be roughly broken into three groups:

- Information about the client, server, and user
- Form data that the user supplied
- Additional pathname information

Most information about the client, server, or user is placed in CGI environment variables. Form data is either incorporated into an environment variable, or is included in the "body" of the request and extra path information is placed in environment variables. Obviously, CGI environment variables are crucial to accessing input to a CGI program

7.1 Using Environment Variables

Much of the most crucial information needed by CGI applications is made available via UNIX environment variables [3]. Programs can access this information as they would any environment variable (e.g., via the *%ENV* associative array in Perl). The following examples demonstrate how environment variables are typically used within a CGI program.

```
#!/usr/local/bin/perl
%list = ('SERVER_SOFTWARE',   'The server
software is: ',
        'SERVER_NAME',        'The server
hostname, DNS alias, or IP address is: ',
        'GATEWAY_INTERFACE', 'The CGI
specification revision is: ',
        'SERVER_PROTOCOL',    'The name and
revision of info protocol is: ',
        'SERVER_PORT',        'The port number
```

```
for the server is: ',
        'REQUEST_METHOD',    'The info
request method is: ',
        'PATH_INFO',         'The extra path info
is: ',
        'PATH_TRANSLATED',   'The translated
PATH_INFO is: ',
        'DOCUMENT_ROOT',     'The server
document root directory is: ',
        'SCRIPT_NAME',       'The script name
is: ',
        'QUERY_STRING',      'The query string
is (FORM GET): ',
        'REMOTE_HOST',       'The hostname
making the request is: ',
        'REMOTE_ADDR',       'The IP address
of the remote host is: ',
        'AUTH_TYPE',         'The authentication
method is: ',
        'REMOTE_USER',       'The
authenticated user is: ',
        'REMOTE_IDENT',      'The remote
user is (RFC 931): ',
        'CONTENT_TYPE',      'The content
type of the data is (POST, PUT): ',
        'CONTENT_LENGTH',    'The length of
the content is: ',
        'HTTP_ACCEPT',       'The MIME types
that the client will accept are: ',
        'HTTP_USER_AGENT',   'The browser
of the client is: ',
        'HTTP_REFERER',      'The URL of the
referer is: ');
print "Content-type: text/html","\n\n";
print "<HTML>", "\n";
print  "<HEAD><TITLE>List of Environment
Variables</TITLE></HEAD>", "\n";
print "<BODY>", "\n";
print "<H1>", "CGI Environment Variables",
"</H1>", "<HR>", "\n";
while ( ($env_var, $info) = each %list ) {
     print $info, "<B>", $ENV{$env_var}, "</
B>", "<BR>","\n";
}

print "<HR>", "\n";
print "</BODY>", "</HTML>", "\n";
exit (0);
```

The associative array contains each environment variable and its description. The *while* loop iterates through the array one variable at a time with the *each* command. Figure 3 shows what the output will look in a browser window.

7.1.1 Displaying information about the server

We look at a simple program that displays various information about the server, such as the CGI and HTTP revisions used and the name of the server software.

```
#!/usr/local/bin/perl
print "Content-type: text/html", "\n\n";
print "<HTML>", "\n";
print "<HEAD><TITLE>About this Server</
TITLE></HEAD>", "\n";
print "<BODY><H1>About this Server</
H1>", "\n";
print "<HR><PRE>";
print "Server Name:      ",
$ENV{'SERVER_NAME'}, "<BR>", "\n";
print "Running on Port:  ",
```

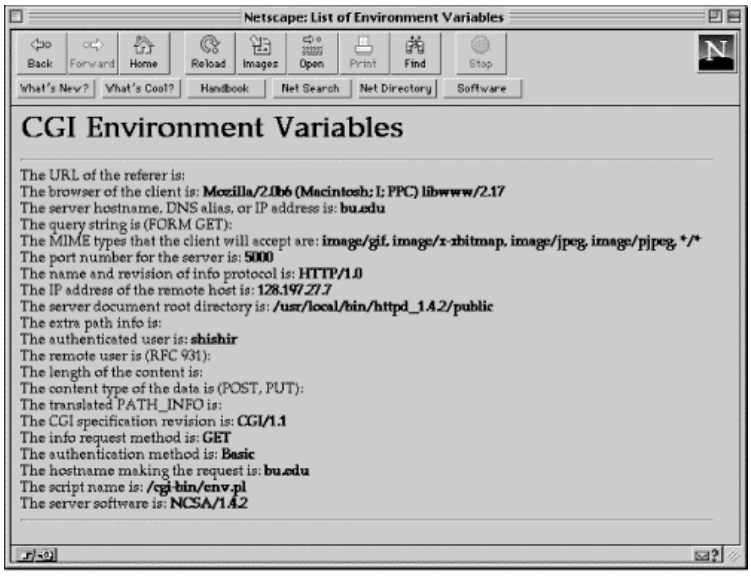


Figure 3: Output of example program

```
$ENV{'SERVER_PORT'}, "<BR>", "\n";
print "Server Software:  ",
$ENV{'SERVER_SOFTWARE'}, "<BR>", "\n";
print "Server Protocol:  ",
$ENV{'SERVER_PROTOCOL'}, "<BR>", "\n";
print "CGI Revision:     ",
$ENV{'GATEWAY_INTERFACE'}, "<BR>",
"\n";
print "<HR></PRE>", "\n";
print "</BODY></HTML>", "\n";
exit (0);
```

going through this program step by step. The first line is very important. It instructs the server to use the Perl interpreter located in the */usr/local/bin* directory to execute the CGI program. Without this line, the server would not know how to run the program, and will display an error stating that it cannot execute the program. Once the CGI script is running, the first thing it needs to generate is a valid HTTP header, ending with a blank line. The header generally contains a content type, also known as a MIME type. In this case, the content type of the data that follows is *text/html*. After the MIME content type is output, we can go ahead and display output in HTML. We send the information directly to standard output, which is read and processed by the server, and then sent to the client for display. Five environment variables are output, consisting of the server name (the IP name or address of the machine where the server is running), the port the server is running on, the server software, and the HTTP and CGI revisions. In Perl, you can access the environment variables through the *%ENV* associative array, keyed by name.

A typical output of this program might look like this:

```
<HTML>
<HEAD><TITLE>About this Server</
TITLE></HEAD>
<BODY><H1>About this Server</H1>
<HR><PRE>
Server Name:      bu.edu
Running on Port:  80
Server Software:  NCSA/1.4.2
Server Protocol:  HTTP/1.0
CGI Revision:     CGI/1.1
<HR></PRE>
</BODY></HTML>
```

7.1.2 To check the Client Browser

One of the more useful items that the server passes to the CGI program is the client (or browser) name. We can put this information to good use by checking the browser type, and then displaying either a text or graphic document. Different Web browsers support different HTML tags and different types of information. If the CGI program generates an inline image, the programmer needs to be sensitive that some browsers support <IMG> extensions that others do not, some browsers support JPEG images as well as GIF images, and some browsers do not support images at all. Using the HTTP_USER_AGENT environment variable, one can determine which browser is being used, and with that information one can fine-tune the CGI program to generate output that is optimized for that browser. Below is a short program that delivers a different document depending on whether the browser supports graphics. We first, identify the browsers that we know do not support graphics. Then get the name of the browser from the HTTP_USER_AGENT variable:

```
#!/usr/local/bin/perl
$nongraphic_browsers = 'Lynx|CERN-
LineMode';
$client_browser  =
$ENV{'HTTP_USER_AGENT'};
```

The variable *$nongraphic_browsers* contains a list of the browsers that do not support graphics. Each browser is separated by the "|" character, which represents alternation in the regular expression we use later in the program. In this instance, there are only two browsers listed, Lynx and *www*. ("CERN-LineMode" is the string the *www* browser uses to identify itself.) The HTTP_USER_AGENT environment variable contains the name of the browser. All environment variables that start with HTTP represent information that is sent by the client. The server adds the prefix and sends this data with the other information to the CGI program.

Now identify the files to return depending on whether the browser supports graphics:

```
$graphic_document = "full_graphics.html";

$text_document = "text_only.html";
```

The variables *$graphic_document* and *$text_document* contain the names of the two documents that we will use. The next thing to do is simply to check if the browser name is included in the list of non-graphic browsers.

```
if ($client_browser =~ /
$nongraphic_browsers/) {

   $html_document = $text_document;

} else {

   $html_document = $graphic_document;

}
```

The conditional checks whether the client browser is one that we know does not support graphics. If it is, the variable *$html_document* will contain the name of the text-only version of the HTML file. Otherwise, it will contain the name of the version of the HTML document that contains graphics.

Finally, print the partial header and open the file. (We need to get the document root from the DOCUMENT_ROOT variable and prepend it to the filename, so the Perl program can locate the document in the file system.)

```
print "Content-type: text/html", "\n\n";
$document_root =
$ENV{'DOCUMENT_ROOT'};
$html_document = join ("/",
$document_root, $html_document);
if (open (HTML, "<" . $html_document)) {
    while (<HTML>) {
        print;
    }
    close (HTML);
} else {
    print "There is a problem with the
configuration on this system!", "\n";
    print "Please inform the Webmaster of the
problem. Thanks!", "\n";
}
exit (0);
```

If the filename stored in *$html_document* can be opened for reading (as specified by the "<" character), the *while* loop iterates through the file and displays it. The *open* command creates a handle, HTML, which is then used to access the file. During the *while* loop, as Perl reads a line from the HTML file handle, it places that line in its default variable *$_*. The *print* statement without any arguments displays the value stored in *$_*. After the entire file is displayed, it is closed. If the file cannot be opened, an error message is output.

7.1.3 Restricting Access for Specified Domains

There are certain information that you would not want everyone on the Internet to see, So you need to set up some type of security to keep the documents away from prying eyes. Say have a set of HTML documents: one for users in your IP domain (e.g., bu.edu), and another one for users outside of the domain. You can also control domain-based access in a CGI script according to what domain the user connects from. The advantage of using a CGI script is that you do not have to turn away other domains, just send them different documents [3]. A look at a CGI program that performs pseudo authentication:

```
#!/usr/local/bin/perl
$host_address = 'bu\.edu';
$ip_address = '128\.197';
```

These two variables hold the IP domain name and address that are considered local. In other words, users in this domain can access the internal information. The period is "escaped" in both of these variables (by placing a "\" before the character), because the variables will be interpolated in a regular expression later in this program. The "." character has a special significance in a regular expression; it is used to match any character other than a newline.

```
$remote_address = $ENV{'REMOTE_ADDR'};
$remote_host = $ENV{'REMOTE_HOST'};
```

The environment variable REMOTE_ADDR returns the IP numerical address for the remote user, while REMOTE_HOST contains the IP alphanumeric name for the remote user. There are times when REMOTE_HOST will not return the name, but only the address (if the DNS server does not have an entry for the domain). In such a case, you can use the following snippet of code to convert an IP address to its corresponding name:

```
@subnet_numbers = split (/\./,
$remote_address);
$packed_address = pack ("C4",
@subnet_numbers);
($remote_host)  = gethostbyaddr
($packed_address, 2);
$local_users = "internal_info.html";
$outside_users = "general.html";
if (($remote_host =~ /\.$host_address$/)
&& ($remote_address =~ /^$ip_address/))
{
$html_document = $local_users;
} else {
$html_document = $outside_users;
}
```

The remote host is examined to see if it ends with the domain name, as specified by the *$host_address* variable, and the remote address is checked to make sure it starts with the domain address stored in *$ip_address*. Depending on the outcome of the conditional, the *$html_document* variable is set accordingly.

```
print "Content-type: text/html", "\n\n";
$document_root =
$ENV{'DOCUMENT_ROOT'};
$html_document = join ("/", $document_root,
$html_document);
if (open (HTML, "<" . $html_document)) {
```

```
    while (<HTML>) {
        print;
    }
    close (HTML);
} else {
    print "There is a problem with the
configuration on this system!", "\n";
    print "Please inform the Webmaster of the
problem. Thanks!", "\n";
}
exit (0);
```

The specified document is opened and the information stored within it is displayed.

7.1.4 User Authentication and Identification

In addition to domain-based security, most HTTP servers also support a more complicated method of security, known as user authentication. When configured for user authentication, specified files or directories are set up to allow access only by certain users. A user attempting to open the URLs associated with these files is prompted for a name and password. The user name and password is checked by the server, and if legitimate, the user is allowed access. In addition to allowing the user access to the protected file, the server also maintains the user's name and passes it to any subsequent CGI programs that are called. The server passes the user name in the REMOTE_USER environment variable. Here is a snippet of code that illustrates what you can do with the REMOTE_USER environment variable:

The HTTP_FROM environment variable also carries information that can be used to identify a user-generally, the user's email address. However, this variable depends on the browser to make it available, and few browsers do, so HTTP_FROM is of limited use.

```
$remote_user = $ENV{'REMOTE_USER'};
if ($remote_user eq "jack") {
    print "Welcome Jack, how is Jack
Manufacturing doing these days?", "\n";
} elsif ($remote_user eq "bob") {
  print "Hey Bob, how's the wife doing? I heard
she was sick.", "\n";
}
.
.
.
```

Server authentication does not provide complete security: Since the user name and password are sent unencrypted over the network, It is possible for a "snoop" to look at this data. For that reason, it is a bad idea to use your real login name and password for server authentication.

7.1.5 Knowing where the remote user came for

Companies who provide services on the Web often want to know from what server (or document) the remote users came. For example, say you visit the server located at http://www.unizik.edu, and then from there you go to http://www.ncs-nig.org/. A CGI program on www.ncs-nig.org can actually determine that you were previously at www.uniziki.edu. This feature is useful for adverts. If a company determines that 90% of all users that visit them come from a certain server, then they can perhaps work something out financially with the webmaster at that server to provide advertising. Also, if your site moves or the content at your site changes dramatically, you can help avoid frustration among your visitors by informing the webmasters at the sites referring to yours to change their links. Here is a simple program that displays this "referral" information:

```
#!/usr/local/bin/perl
print "Content-type: text/plain", "\n\n";
$remote_address = $ENV{'REMOTE_ADDR'};
$referral_address = $ENV{'HTTP_REFERER'};
print "Hello user from $remote_address!", "\n";
print "The last site you visited was:
$referral_address. Am I genius or what?", "\n";
exit (0);
```

The environment variable HTTP_REFERER, which is passed to the server by the client, contains the last site the user visited before accessing the current server. There are three important things you need to remember before using the HTTP_REFERER variable:

- First, not all browsers set this variable.
- Second, if a user accesses your server first, right at startup, this variable will not be set.
- Third, if someone accesses your site via a bookmark or just by typing in the URL, the referring document is meaningless. So if you are keeping some sort of count to determine where users are coming from, it would not be totally accurate.

7.2 Accessing Form Input

In this section, we briefly look at the basic concept behind forms, a more detailed explanation can be obtained via literature. Forms provide a way to get input from users and supply it to a CGI program, as shown in Figure 4 The Web browser allows the user to select or type in information, and then sends it to the server when the Submit button is pressed.

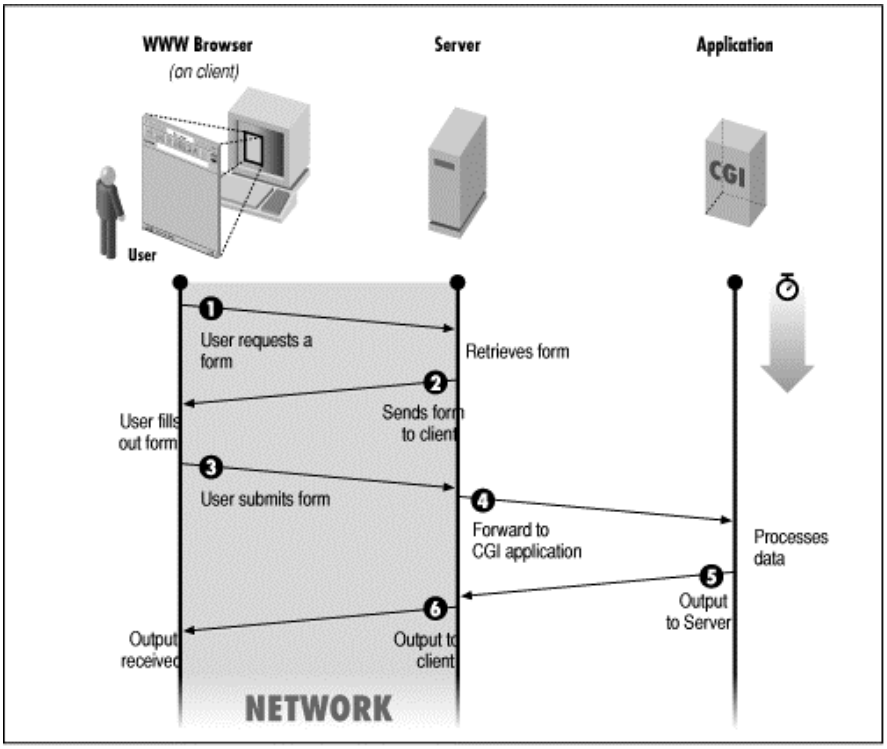


Figure 4: Form interaction with CGI

7.3 Query Strings

One way to send form data to a CGI program is by appending the form information to the URL, after a question mark. You may have seen URLs like the following:

http://some.machine/cgi-bin/name.pl?fortune

Up to the question mark (?), the URL should look familiar. It is merely a CGI script being called, by the name name.pl. What is new here is the part after the "?". The information after the "?" character is known as a *query string*. When the server is passed a URL with a query string, it calls the CGI program identified in the first part of the URL (before the "?") and then stores the part after the "?" in the environment variable QUERY_STRING. The following is a CGI program called name.pl that uses query information to execute one of three possible UNIX commands.

```
#!/usr/local/bin/perl
print "Content-type: text/plain", "\n\n";
$query_string = $ENV{'QUERY_STRING'};
if ($query_string eq "fortune") {
    print '/usr/local/bin/fortune';
} elsif ($query_string eq "finger") {
    print '/usr/ucb/finger';
} else {
    print '/usr/local/bin/date';
}
exit (0);
```

You can execute this script as either:

```
http://some.machine/cgi-bin/name.pl?fortune
http://some.machine/cgi-bin/name.pl?finger
```

or

```
http://some.machine/cgi-bin/name.pl
```

and you will get different output. The CGI program executes the appropriate system command (using backtics) and the results are sent to standard output. In Perl, you can use backtics to capture the output from a system command. When executing any type of system commands in CGI applications, the user should be very careful because of possible security problems. Never do something like this:

```
print '$query_string';
```

The danger is that a diabolical user can enter a dangerous system command, such as:

```
rm -fr /
```

which can delete everything on your system. Nor should you expose any system data, such as a list of system processes, to the outside world.

7.4 A Simple Form

The following example is a more realistic illustration of how forms work with CGI. Instead of supplying the information directly as part of the URL, we will use a

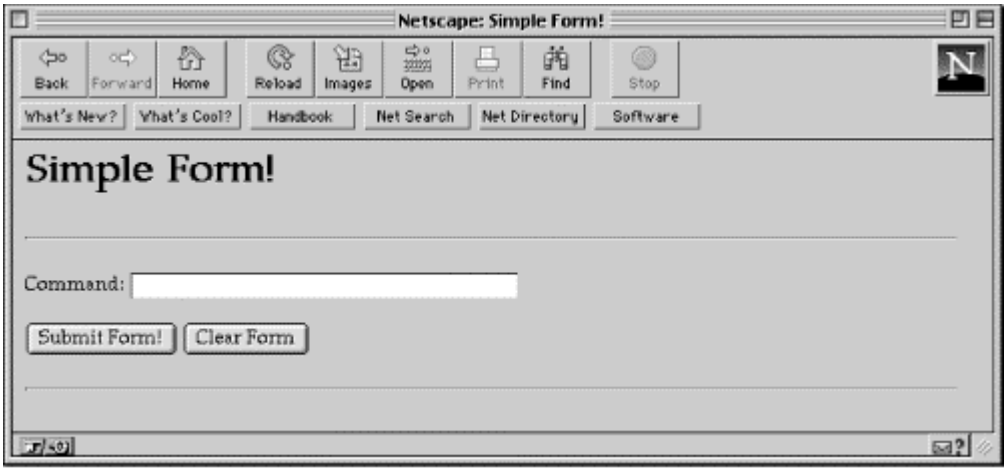


Figure 5: Simple form in Netscape

form to solicit it from the user.

```
<HTML>
<HEAD><TITLE>Simple Form!</TITLE></
HEAD>
<BODY>
<H1>Simple Form!</H1>
<HR>
<FORM ACTION="/cgi-bin/unix.pl"
METHOD="GET">
Command: <INPUT TYPE="text"
NAME="command" SIZE=40>
<P>
<INPUT TYPE="submit" VALUE="Submit Form!">
<INPUT TYPE="reset" VALUE="Clear Form">
</FORM>
<HR>
</BODY>
</HTML>
```

Since this is HTML, the appearance of the form depends on what browser is being used. Figure 5 shows what the form looks like in Netscape.

This form consists of one text field titled "Command:" and two buttons. The Submit Form! button is used to send the information in the form to the CGI program specified by the ACTION attribute. The Clear Form button clears the information in the field.

The METHOD=GET attribute to the <FORM> tag in part determines how the data is passed to the server. We'll talk more about different methods soon, but for now, we'll use the default method, GET. Now, assuming that the user enters "fortune" into the text field, when the Submit Form! button is pressed the browser sends the following request to the server:

GET /cgi-bin/unix.pl?command=fortune HTTP/1.0

.

. (header information)

.

The server executes the script called *unix.pl* in the cgi-bin directory, and places the string "command=fortune" into the QUERY_STRING environment variable. Think of this as assigning the variable "command" (specified by the NAME attribute to the <INPUT> tag) with the string supplied by the user, "fortune".

command=fortune

Let's go through the simple *unix.pl* CGI program that handles this form:

```
#!/usr/local/bin/perl

print "Content-type: text/plain", "\n\n";

$query_string = $ENV{'QUERY_STRING'};

($field_name, $command) = split (/=/, $query_string);
```

After printing the content type (*text/plain* in this case, since the UNIX programs are unlikely to produce HTML output) and getting the query string from the *%ENV* array, we use the *split* function to separate the query string on the "=" character into two parts, with the first part before the equal sign in *$field_name*, and the second part in *$command*. In this case, *$field_name* will contain "command" and *$command* will contain "fortune." Now, we're ready to execute the UNIX command:

```
if ($command eq "fortune") {
    print '/usr/local/bin/fortune';
} elsif ($command eq "finger") {
    print '/usr/ucb/finger';
} else {
```

```
    print '/usr/local/bin/date';
}
exit (0);
```

Since we used the GET method, all the form data is included in the URL. So we can directly access this program without the form, by using the following URL:

http://some.machine/cgi-bin/
unix.pl?command=fortune
It will work exactly as if you had filled out the form and submitted it.

The GET and POST Methods

Another method that can be used to process forms is called POST. Using the POST method, the server sends the data as an input stream to the program. That is, if in the previous example the <FORM> tag had read:

```
<FORM ACTION="unix.pl" METHOD="POST">

the following request would be sent to the server:

POST /cgi-bin/unix.pl HTTP/1.0

.

. (header information)

.

Content-length: 15

command=fortune
```

The version of *unix.pl* that handles the form with POST data follows. First, since the server passes information to this program as an input stream, it sets the environment variable CONTENT_LENGTH to the size of the data in number of bytes (or characters). We can use this to read exactly that much data from standard input.

```
#!/usr/local/bin/perl

$size_of_form_information =
$ENV{'CONTENT_LENGTH'};
```

Second, we read the number of bytes, specified by *$size_of_form_information*, from standard input into the variable *$form_info*.

```
read (STDIN, $form_info,
$size_of_form_information);
```

Now we can split the *$form_info* variable into a *$field_name* and *$command*, as we did in the GET version of this example. As with the GET version, *$field_name* will contain "command," and *$command* will contain "fortune" (or whatever the user typed in the text field). The rest of the example remains unchanged:

```
($field_name, $command) = split (/=/,
$form_info);
print "Content-type: text/plain", "\n\n";
if ($command eq "fortune") {
    print '/usr/local/bin/fortune';
} elsif ($command eq "finger") {
    print '/usr/ucb/finger';
} else {
    print '/usr/local/bin/date';
}
exit (0);
```

Since it's the form that determines whether the GET or POST method is used, the CGI programmer can not control which method the program will be called by. So scripts are often written to support both methods. The following example will work with both methods:

```
#!/usr/local/bin/perl
$request_method =
$ENV{'REQUEST_METHOD'};
if ($request_method eq "GET") {
        $form_info = $ENV{'QUERY_STRING'};
} else {
     $size_of_form_information =
$ENV{'CONTENT_LENGTH'};
     read (STDIN, $form_info,
$size_of_form_information);
}
($field_name, $command) = split (/=/,
$form_info);
print "Content-type: text/plain", "\n\n";
if ($command eq "fortune") {
    print '/usr/local/bin/fortune';
} elsif ($command eq "finger") {
    print '/usr/ucb/finger';
} else {
    print '/usr/local/bin/date';
 }
exit (0);
```

The environment variable REQUEST_METHOD contains the request method used by the form. In this example, the only new thing we did was check the request method and then assign the *$form_info* variable as needed.

7.5 Extra Path Information

Besides passing query information to a CGI script, you can also pass additional data, known as extra path information, as part of the URL. The extra path information depends on the server knowing where the

name of the program ends, and understanding that anything following the program name is "extra." Here is how you would call a script with extra path information:

```
http://some.machine/cgi-bin/display.pl/cgi/
cgi_doc.txt
```

Since the server knows that *display.pl* is the name of the program, the string "/cgi/cgi_doc.txt" is stored in the environment variable PATH_INFO. Meanwhile, the variable PATH_TRANSLATED is also set, which maps the information stored in PATH_INFO to the document root directory (e.g., */usr/local/etc/httpd/ public/cgi/cgi-doc.txt*).

Here is a CGI script—*display.pl*—that can be used to display text files located in the document root hierarchy:

```
#!/usr/local/bin/perl
$plaintext_file = $ENV{'PATH_TRANSLATED'};
print "Content-type: text/plain", "\n\n";
if ($plaintext_file =~ /\.\./) {
   print "Sorry! You have entered invalid
characters in the filename.", "\n";
      print "Please check your specification and
try again.", "\n";
} else {
   if (open (FILE, "<" . $plaintext_file)) {
      while (<FILE>) {
         print;
      }
      close (FILE);
   } else {
      print "Sorry! The file you specified cannot
be read!", "\n";
        }
}
exit (0);
```

In this example, we perform a simple security check. We make sure that the user didn't pass path information containing "..". This is so that the user cannot access files located outside of the document root directory. Instead of using the PATH_TRANSLATED environment variable, you can use a combination of PATH_INFO and DOCUMENT_ROOT, which contains the physical path to the document root directory. The variable PATH_TRANSLATED is equal to the following statement:

```
$path_translated = join ("/",
$ENV{'DOCUMENT_ROOT'},
$ENV{'PATH_INFO'};
```

However, the DOCUMENT_ROOT variable is not set by all servers, and so it is much safer and easier to use PATH_TRANSLATED.

7.6 Output from the Common Gateway Interface

CGI programs are requested like any other regular documents. The difference is that instead of returning a static document, the server executes a program and returns its output. As far as the browser is concerned, however, it expects to get the same kind of response that it gets when it requests any document, and it's up to the CGI program to produce output that the browser is comfortable with. Web. However, there are other things you can do, such as:

- Return graphics and other binary data
- Tell the browser whether to cache the virtual document
- Send special HTTP status codes to the browser
- Tell the server to send an existing document

Each of these techniques involves knowing a little bit about returning additional headers from the CGI program. CGI applications can return nearly any type of virtual document, as long as the client can handle it properly. It can return a plain text file, an HTML file … or it can send PostScript, PDF, SGML, etc. The only one type of header that CGI programs can use is the "Content-type" header. "Content-type" is an HTTP header that contains a MIME content type describing the format of the data that follows. Other headers can describe:

- The size of the data
- Another document that the server should return (that is, instead of returning a virtual document created by the script itself)
- HTTP status codes

7.7 Server Redirection

Another thing CGI programs can do is to instruct the server to retrieve an existing document and return that document instead. This is known as *server redirection*. To perform server redirection, you need to send a *Location* header to tell the server what document to send. The server will retrieve the specified document from the Web, giving the appearance that the client had *not* requested your CGI program, but that document. A common use for this feature is to return a generic document that contains static information. For example, say you have a form for users to fill out, and you want to display a thank-you message after someone completes the form. You can have the CGI program create and display the message each time it is called. But a more efficient way would be for the program to

send instructions to the server to redirect and retrieve a file that contains a generic thank-you message.

Suppose you have an HTML file like the one below (thanks.html), that you want to display after the user fills out one of your forms:

```
<HTML>
<HEAD><TITLE>Thank You!</TITLE></HEAD>
<BODY>
<H1>Thank You!</H1>
<HR>
Thank You for filling out this form. We will be using
your
input to improve our products.
Thanks again,
WWW Software, Inc.
</BODY>
</HTML>
```

You could use the programs discussed earlier to return static documents, but it would be counterproductive to do it in that manner. Instead, it is much quicker and simpler to do the following:

```
#!/usr/local/bin/perl
print "Location: /thanks.html", "\n\n";
exit (0);
```

The server will return the HTML file *thanks.html* located in the document root directory. You do not have to worry about returning the MIME content type for the document; it is taken care of by the server [3]. An important thing to note is that you cannot return any content type headers when you are using server redirection. You can use server redirection to your advantage and design CGI applications like the following:

```
#!/usr/local/bin/perl
$uptime = '/usr/ucb/uptime';
($load_average) = ($uptime =~ /average: ([^,]*)/
);
$load_limit = 10.0;
$simple_document = "/simple.html";
$complex_document = "/complex.html";
if ($load_average >= $load_limit) {
      print "Location: $simple_document", "\n\n";
} else {
     print "Location: $complex_document", "\n\n";
}
exit (0);
```

This program checks the load average of the host system with the *uptime* command. Depending on the load average, one of two documents is returned; a rich, complicated HTML document with graphics if the system is not "busy," or a simple text-only document otherwise. And the last thing to note is that you are not limited to returning documents on your own server. You can also return a document (static or virtual) located elsewhere on the Internet, so long as it has a valid URL:

```
print "Location: http://www.ncs-nig.org", "\n\n";
```

for example, this statement will return the home page for NCS.

7.8 Server Side Includes

Server Side Includes are directives which the programmer can place into the HTML documents to execute other programs or output such data as environment variables and file statistics [3]. Unfortunately, not all servers support these directives; However, there is a CGI program called *fakessi.pl* that can be used to emulate Server Side Includes if one's server does not support them. While Server Side Includes technically are not really CGI, they can become an important tool for incorporating CGI-like information, as well as output from CGI programs, into documents on the Web. When the client requests a document from the SSI-enabled server, the server parses the specified document and returns the evaluated document (see Figure 6). The server does not automatically parse all files looking for SSI directives, but only ones that are configured as such.

SSI does have its disadvantages. First, it can be quite costly for a server to continually parse documents before sending them to the client. And second, enabling SSI creates a security risk. Novice users could possibly embed directives to execute system commands that output confidential information. Despite these shortcomings, SSI can be a very powerful tool if used cautiously.

7.9 Conclusion

The **C**ommon **G**ateway **I**nterface is a specification for transferring information between a World Wide Web server and a CGI program. CGI is a generic interface for calling external programs to crunch numbers, query databases, generate customized graphics, or perform any other server-side task. A CGI program is any program designed to accept and return data that conforms to the CGI specification. The program could be written in any programming language, including C, Perl, Tcl, Java or Visual Basic [2], [3], [5]. The hardest aspect of developing CGI applications on the Web is the testing/debugging phase [4]. The main reason for the difficulty is that applications are being run across a network, with client and server interaction. When there are errors in CGI programs, it is difficult to figure out

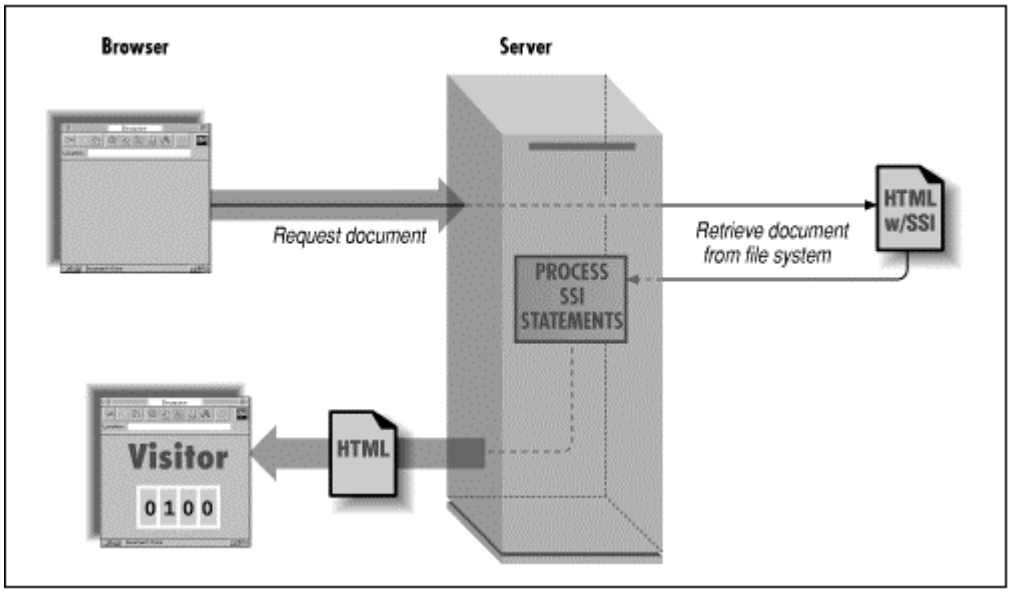


Figure 6: Server Side Includes

where they lie. One problem with CGI is that each time a CGI script is executed, a new process is started [2]. For busy web sites, this can slow down the server noticeably. A more efficient solution, but one that it is also more difficult to implement, is to use the server's API, such as ISAP or NSAPI. Another increasingly popular solution is to use Java servelets.

References

1. Boutell, T(2003) Creating Web sites. **http://www.boutell.com/newfaq/.** Downloaded, March, 2005

2. Castro E. (1999) Perl and CGI on the World Wide Web: Berkeley, CA : Peachpit Press.

3. Gundavaram S. (1996). CGI programming on the World Wide Web Bonn Publishers, Cambridge

4. Nick Kew, CGI Programming FAQ, Copyright © Nick Kew, 1996-2000. http://www.htmlhelp.org/faq/cgifaq.html, http://www.webthing.com/tutorials/cgifaq.html

5. Tittel E. (1996) Web programming Secrets with HTML, CGI and Perl . Foster City, CA : IDG Books Worldwide.

5. Tittel, E. (1995) Foundations of CGI programming on the World Wide Web Foster City, CA : IDG Books Worldwide, USA

6. Colburn R. (1998) SAMS' Teach yourself CGI programming in a week Sams.net Pub. Indianapolis, IN

Architectural Design and Implementation of E-Crm for Web Solution

Oke, Akinniran Oluwagbohun, Ajagbe, Temitope Samuel

ABSTRACT

This paper presents an integrated framework for modular design and phased implementation of Customer Relationship Management (CRM) solution. Research programs in information system have classified five e-CRM domains namely e-CRM within markets, e-CRM in Business Models, Knowledge Management for e-CRM, e-CRM Technological issues, e-CRM on Human Issues. To avoid the general pitfall of bogus implementation strategy, we carefully outlined a build-test-and-commit-or-rollback plan for both mid-sized and large enterprises presently in Nigeria, - Africa as a whole. E-CRM within markets addresses how companies adopt the use of e-CRM design. This project will provide a framework for companies in Nigeria who are just detaching from the monolith-architecture of customer relationship management, thus increasing their profit in the short and long run.

1.0 Overview

In today's highly competitive and volatile marketplace, it is critical for companies to have the ability to detect and respond in a timely manner to changing customer needs across all contact points, introduce new offerings and understand the competition. It is equally important to have timely insight into one's own companies. Not being aware of the state of your sales pipeline, effectiveness of marketing campaigns or the quality of customer service can be detrimental to the success of a company as they impact future expenditures on everything ranging from marketing, product development and capital equipment.

CRM applications have been built to meet this challenge but the cost of implementation and the failure rate of getting such projects to successful rollout have been questionable.

Investments in CRM process changes can be made incrementally**,** in small pieces. In some ways, the move toward incremental or modular rollout of customer initiatives was part of the natural maturation of CRM in general. Although to be effective a customer initiative must have enterprise-wide impact, emergent pilot programs, specific programs designed to address aching points or capture "low-hanging ROI issues" and the increasing level of sophistication of CRM called for a piecemeal approach.

The down economy also played a role in moving toward incremental rollout. In contrast to many poorly implemented big bang approaches to CRM of the late 1990s in which technology was viewed as a "silver bullet," modular approaches designed to generate ROI proof points became necessary to justify continued investment of time, money and employee resources spent on it. As these modular initiatives began to deliver, the returns and the intelligence from each incremental step could be rolled up into enterprise-wide programs and sustainable ROI.

CRM Model Features

- Sales Force Automation
- Marketing Automation
- Customer Service
- Profiling and Segmentation
- Community Tools
- Personalization

Implementing these model features in a single project may have sacrificed the quality and focus that is required for a well-managed requirement.

CRM solutions can be broadly categorized into three areas namely

- Operational CRM
- Analytical CRM
- Collaborative CRM

Operational CRM supports the front office process and supervision [6]. An example of this reflects a staff in a

call center where complaints of customers and different contact are being obtained. Operational CRM deals with the *day-to-day running* of a business or organization.

Analytical CRM builds on operational CRM and establishes information on customer segments, behaviors and value using statistical methods [7], [8]. This has to do with the sales management.

Collaborative CRM concentrates on customer integration using a coordinated mix of interaction channels (multi - channel management) [6]. This particular CRM can be regarded as a hybrid of Analytical and Operational CRM. This acts of the two former CRM. It deals with management of the two.

1.1 Justification

In Nigeria, leading organization such as Shell Oil Company is currently deploying the SAP for it B2B2C market analysis. Globacom is the only telecommunication who is presently considering the adoption of CRM (Siebel product) for its B2C Business. It must be emphasized that most of this CRM build outside Nigeria culture must be customized to meet our needs internally. Also, most of these CRM takes a lot of revenue to implement. This is one of the issues while CRM products are not being used by most companies in Nigeria. Having this in mind, there is a need to build a scalable and cultural CRM product for Nigeria companies and for Africa as a whole.

2.0 Incremental Rollout, Incremental Gains

The economy has created unprecedented pressure on many organizations to show a quantitative return on their investments, whether that return is higher revenue, greater market share or improved profitability.

Short-term wins that lead to long-term ROI have become critical factors in justifying investment to shareholders, partners, management and even employees. Recent studies suggest that this modular approach is paying off. A survey by AMR Research reports that 19 percent of companies will be investing in modular, customer-based technology in 2003, based on short-term business initiatives and economic returns. This concurs with research from Giga Information Group saying 2003 marks a shift in strategic thinking toward a focus on smaller, more incremental customer-based projects.

A growing number of companies are realizing that investments in CRM do not have to be all-or-nothing technological or cultural upheavals. In fact, most CRM Initiatives include some sort of customer-specific pilot program or alternatively, a means of getting more from under-performing current technology. Test, learn, evolve and improve: this is an approach to customer strategy can build long-term strategic advantage.

For example, in 2002, one nationally known, mid-market e-services provider began to look more closely at its customer interactions. Rather than purchase a vast CRM system, the company hired an outside provider as a third party to identify, gather, interpret and disseminate actionable relationship data.

Working together to scour the principal points of customer contact, the e-company and its third-party provider were able to identify a handful of critical Relationship drivers. From there, the company developed a scorecard and began surveying its customers in order to establish a baseline “temperature” for the quality of each individual relationship. Now, relying on these baseline scores, the company measures the impact of incremental customer initiatives not only on overall sales, but also on individual activity and satisfaction. According to a company Vice President, “The early ROI data is very positive.” A second example of a smart incremental investment paying off is the case of one of the 10 largest trucking companies in the U.S. In this instance, benefits came in the forms of financial stability and insulation from pricing pressure. The trucking giant was finding it increasingly difficult to maintain its rates, as well as weather cost increases in fleet insurance, fuel and maintenance. The company looked to increase the value of its customer relationships in order to stabilize-and even raise-its rates, while simultaneously reducing service costs.

The call center was the principal point of contact between the company and its customers. Historically calls were directed to designated agents, who each managed three lines. The result: Customers often found themselves “on hold” waiting for *their* agent. Even more damaging, the company was unable to efficiently prioritize call handling because it didn’t know who its MVCs were. An incremental investment in call-center technology created major improvements in customer service.

The new system provided call-center representative with more complete customer information, including prioritizing calls according to customer value. Representatives are now able to handle customer calls more efficiently and “hold time” has decreased from 41 seconds to 4 seconds. At the same time, the call-abandon rate has plummeted to 2.6 percent, from 17 percent. Less-profitable customers have been migrated to a self-service Web site. Customers can now use the site to request rates, measure mileage between delivery

points, book freight loads, track their deliveries and request proof of delivery.

The improved efficiency of its customer service has allowed the company to decrease call-center costs by $1 million annually, with an additional $1 million in savings on phone bills. Perhaps more importantly, the improvements in customer service have created value for the customer to the point where the company is experiencing less resistance to rate increases. Revenue has climbed nearly 18 percent, even as its employee head count remained constant.

3.0 Architecture Designs

Architecture is a set of structuring principles that enables a system to be comprised of a set of simpler systems each with its own local context that is independent of but not inconsistent with the context of the larger system as a whole[5].

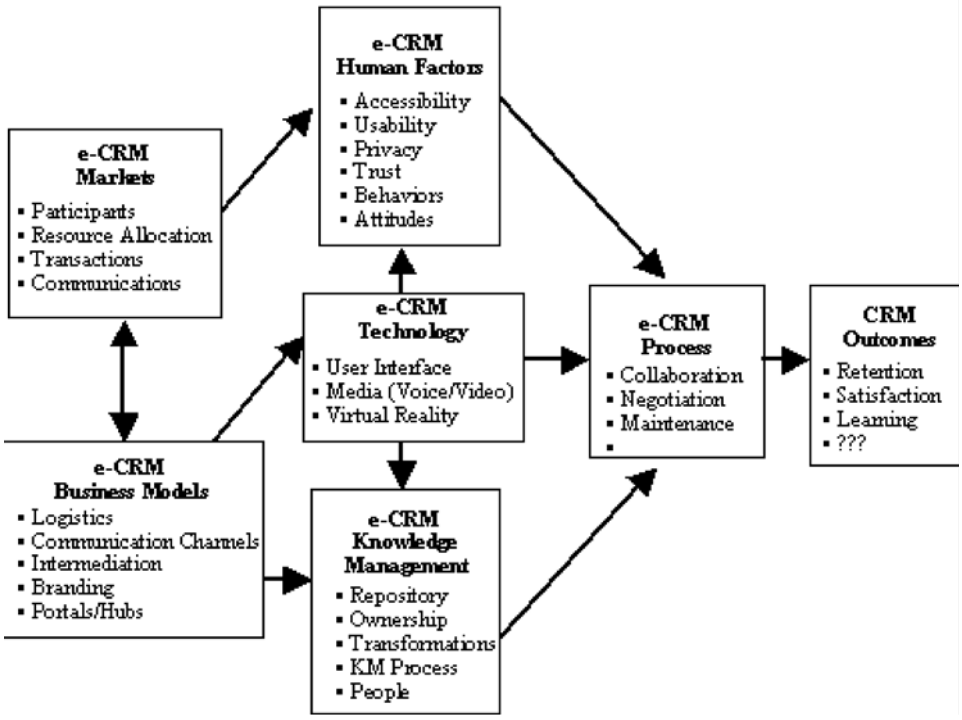
As you create an architecture to satisfy the business and service-level requirements of a system, you usually do not have unlimited funds to purchase hardware, software and development resources, so there is a need to make the system work within your predefined limitations. Customer-service excellence would be the standard by which a company measured its success with measurable time i.e it must be time bounded.

Building and maintaining customer relationship is neither new nor necessaryly tied to the use of information technology. Nonetheless, the use of customer relationship management (CRM) system is becoming increasingly important to improve customer lifetime value. It is a strategy used to learn more about customers' needs and behaviors in order to develop stronger relationships with them.

3.1 System Design Model and Data Warehouse model

For a successful CRM to be developed, models involved in the enterprise software development must be considered. According to Todman [11], traditional development system model approach and new incremental system development dimensional model approach are the two major developing techniques for CRM data warehouse. Below is the model variable for proper development of e-CRM solution for organizations. This entire model has lot of internal relating models.

Figure 1



Source: Romano & Fjermestad / An Agenda For ECCRM Research, 2001

Figure 1 presents a model for the study of e-CRM. We define five input variables: Markets, Business Models, Knowledge Management, Technology, and Human Factors. These general categories are consistent with models presented in other areas of IS research such as Group Support Systems. Together these input variables define both the human potential and the technological infrastructure for developing e-CRM processes to achieve valuable outcomes. The e-CRM process is continuous and evolutionary and consists of organizational members and individuals from outside the organization using e-CRM technologies to establish, develop, and maintain important successful customer relationships, which are the outcomes of relevant e-CRM processes. Within each component of the model an illustrative list of example concepts is provided. We believe that these are *some* of the most important areas for research in e-CRM within these five fundamental areas

3.2 E-CRM Technology

Data warehousing design and implementation for e-CRM is now a mature business solution which must be taken into perspective for effective design of CRM, thus, the evolution of business requires the evolution of scalable data warehouses. That business people need to grasp the CRM nettle is an absolute fact. In order to do this, information is the key. It is fair to say that an organization cannot be successful at CRM without high-quality, timely, and accurate information.

CRM data warehouse has two major development approaches, one of this design approach is the old water fall process approach called **Traditional development** approach and **Incremental system** approach.

Traditional development system model approach points out that system must be developed in a hierarchical order. This system development lifecycle that was adopted consisted of a set of major steps:

Requirements gathering

- Systems analysis
- System design
- Coding
- System testing
- Implementation.

The problem with this approach was that each step had to be completed before the next could really begin.

Incremental system development approach illustrates that development stages within the former approach should be considered all together with the business analyst working in hand with them. It also states that it must consider feedback from the customers who needs the solution should model the solution. Since the whole of CRM is customer driven solution. This approach can also be called Rapid application development (RAD) and it involves piecemeal development. These steps involved are:

- Pilot implementation
- Quick wins
- Prioritization

Incremental approach helps in understanding various set of CRM and this will help with the design of data warehouse. It also helps in reducing failure of CRM products and allows realization of targets by reducing customers churning. Even, statistic shows that for a full-grown CRM warehouse to be achieved, it must take a full 2 years of data development and building up for customers' detailed data to be obtained. It also helps in modeling out ***customer insight***. Customer insight is what makes us to be able to predict customer behaviors and circumstances. This helps us in differentiating the customers into different segment, thus postulating different business strategy in retention of those customers.

The first-generation data warehouses tended to focus on the analysis of behavioral information. Well, the second generation needs to support big business issues such as CRM and, in order to do this effectively; we have to be able to focus not only on behavior, but circumstances as well.

3.3 Segmentation of Customers

The segmentation of customers helps in categorizing groups called customers based on the information entered into the design view forms. This information is about the customers. This category can be based on

- Customer's circumstance
- Behavior circumstance
- Derived circumstance

These entire categories are obtained from the time bounded information services management to analytical information and they generate to predicting customers insight. Having the customer insight will enable us to predict the time of product needed by the customers, the product versus time curve will be determined on time so that the company can give better projections.

This will also reduce unnecessary cost that may be incur. Knowing the customer insight will help in increasing the lifetime value of customer with the company. It also increases the up-selling and cross-selling potential of products. It shows the entanglement potential inherent in customers. It also reduces churn rate.

All companies have a churn metric no of customer lost percentage compared to the percentage of initial state of business.

3.4 Proposition Methodology

In this paper, incremental system implementation is adopted. Implementation of CRM system can come in 3 to 4 major modules steps. The principal criteria for a uniform presentation to the customer ("one face to the customer") are on the one hand the assurance of the same quality over all contact points (e.g. call center, mail, fax, Internet, field representatives). On the other hand customer information has to be spread throughout the organization (integration of the front and back offices). These modules are:

- Sales and Marketing Management
- Services Management
- Analytical CRM (Management CRM)

In designing these 3 modules, each of the modules contains internal sub-modules. In modeling the internal relations, the business process must be taken into consideration. This business processes relationship must be modeled into a development process in Unified Modeling Language (UML) languages. This helps in arising at the object relationship between the business entities using enterprise JavaBeans controller for the manipulation of the objects. A model is a simplification of reality. Building models using UML diagrams will help to better understand the system that is being developed in:

- Visualize the system as it is.
- Specify the structure or behavior of a system
- Provide a template to guide the construction of the system
- Document the decisions made about the system

In each of these models, contact management, campaign management, insight mining, current and historical account details activities interrelated to each other. Reporting of all using graphical means is also considered. There are other sub-models which will arise from this. The **master block** in figure 2 shows where the segmentation is taking place. Each block generate to models defining detail CRM. Considering the model, these blocks stages generates to prospecting of customers, lead qualification, workflow, demonstration of pipelines and quotes with the sales forecasting in e-CRM solution. See Figure 2.

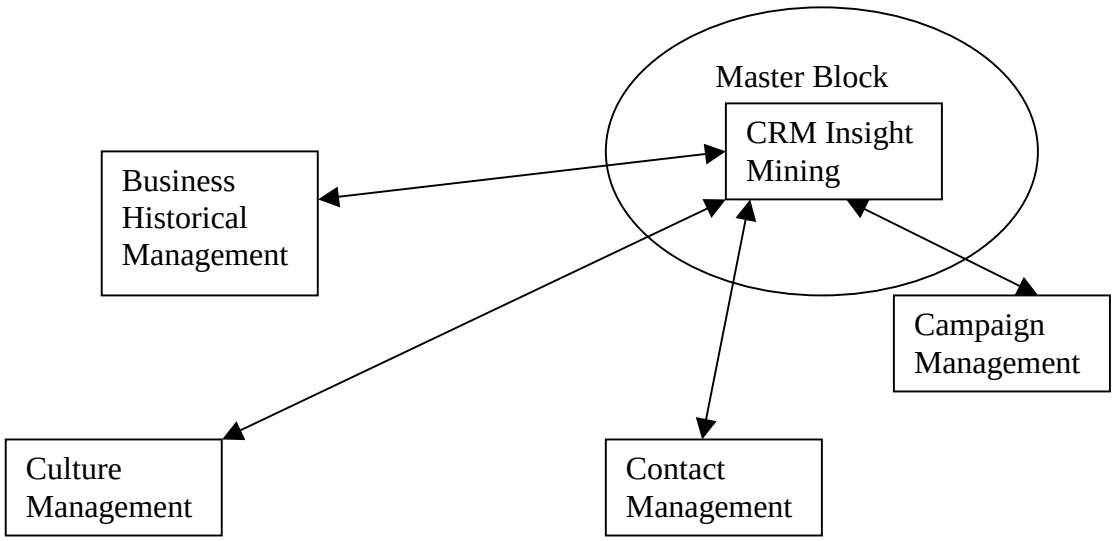


Figure 2: CRM Model

4.0 Implementation and Result

Worldwide, it is obvious that the gains realized from a customer-focused strategy are attracting companies of all sizes to implement a CRM System. According to Framingham, Mass.[12] -based IDC forecasts that spending in the CRM industry will increase at a compound annual growth rate (CAGR) of 25% by the year 2005 ($148 billion from $61 billion in 2001). Most of the leading CRM companies have used these forecasting projections in investing into CRM implementations. E.g Siebel. Let alone this projections, the cost of implementation of CRM and the tool to use is one of the factors that affect the CRM project design. In this design and implementation, and based on research and worldwide benchmarking, it has being found that ARIS toolset is one of the best implementation tool for business. This tool is used in implementing Siebel (Siebel Reference Model) CRM edge-cutting solution which is customer oriented and SAP R/3 Reference Model which is B2B2C.

For better implementation of CRM, the design tool and features in modeling data warehouse must be taken into consideration. Business Process Design is a three-stage process by which companies can tailor their business processes to their own requirements and needs and to those of the market [13]. This element of the continuous improvement cycle comprises the three aspects of design, analysis and optimization.

Considering the graph below for (A) Manufacturing Company, it can be seen that the results generated for the use of e-CRM is very encouraging. Comparing the result obtained with the use of eCRM and without the use of eCRM, the ROI generated for (A) manufacturing company without the use of eCRM is unpredictable. The reasons for this can be from the increase in acquisition cost, retention cost and administrative cost. Increase in all these costs will lower the ROI obtained for the company at the end of the year. This is depicted in the diagrams below. If the sum of these costs is greater than the gross profit, the company will eventually run at a loss. This is shown in figure 3.1, 3.2, 3.3. With eCRM on the manufacturing company, the result generated is predictable and it keeps increasing thus bringing better dividend for the investors. This also results in better ROI prediction. The eCRM strategy helps in reducing the marketing expenses, distribution expenses, administrative expenses, and acquisition expenses. The retention expense for one customer is reduced. When all these are done using eCRM, better ROI can be obtained. This is also shown in figure 3.4, 3.5, 3.6.

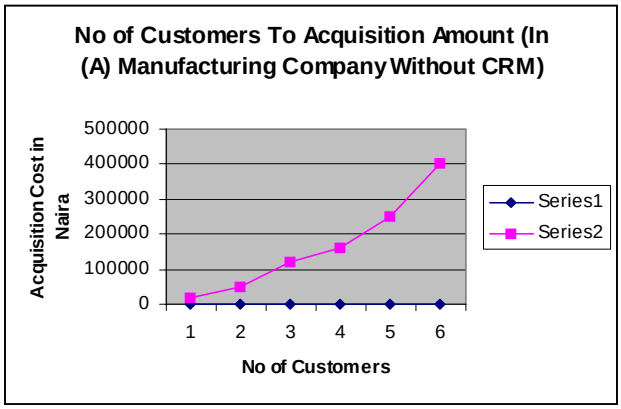


Figure 3.1: Number of Customers in relation to Acquisition Cost of a (A) Manufacturing Company.

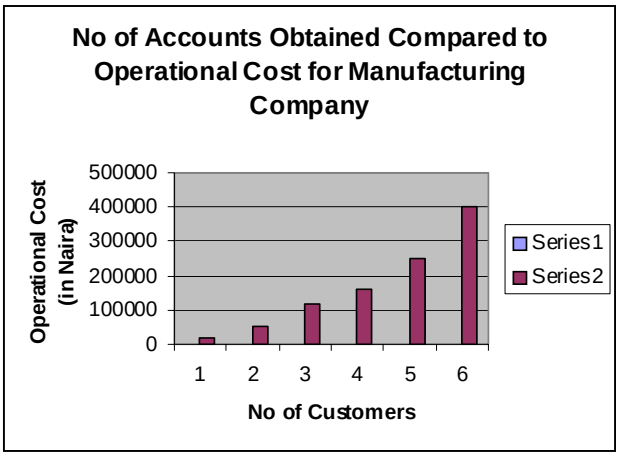


Figure 3.2: No of Customers obtained in relation to Operational Cost of a (A) Manufacturing Company.

Figure 3.3: Year obtained in relation to ROI Amount of a (A) Manufacturing Company.

Figure 3.5: Retention Cost to Increase in Customer of a (A) Manufacturing Company. (With CRM)

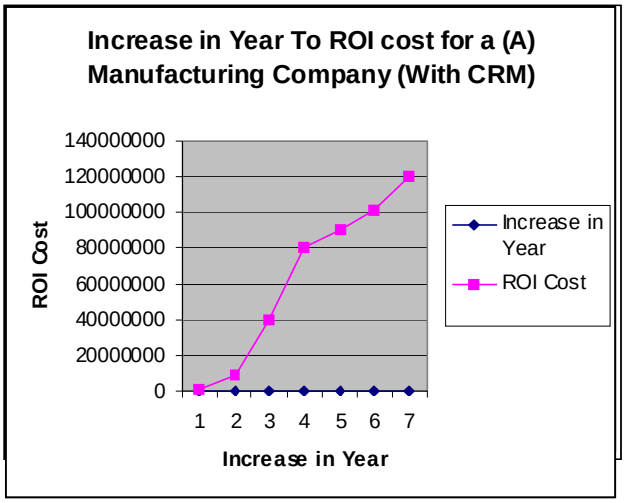


Figure 3.4:Number of Customers in relation to Acquisition Cost of a (A) Manufacturing Company (with CRM)

Figure 3.6 Year compared to ROI Amount of a (A) Manufacturing Company. (with CRM)

Conclusion

The execution of such customer-oriented strategies, in turn, becomes a strategic capability for the organization and a competitive advantage in a down economy. Companies that have intimate customer knowledge can both predict and react to changes in customer behavior more quickly and efficiently than competitors. An organization that implements an incremental rollout of CRM solution, selecting adequate CRM category will guarantee steady Return On Investment.

Finally, eCRM must come to stay in our companies if we want to match up to the world best companies' strategies. For any company investing in e-CRM, figure 3.6 can otherwise be termed as the forecasting projections expected. One of such companies is Cisco Corporation. Cisco gained some $450 million annually in cost savings by moving many of its supply chain operations and customer relationship management to the Internet, making it feasible for its suppliers to access the company's enterprise resource planning system. This permits Cisco and its suppliers to see product demand and product planning on a real-time basis; it also makes efficiency processes feasible (e.g., dynamic replenishment), enabling the company to reduce its respective inventories in a significant way, and to do so without compromising production processes and product availability.

References

[1] Keen, P.G.W. Competing of internet business, Eburon Publishers: Delft, The Netherlands., 1999.

[2] Chatham, B.; Orlov, L.M.; Howard, E.; Worthen, B., and Coutts, A. *The Customer Conversation.* Cambridge: Forrester Research, Inc., 2000.

[3] Thompson, E. CRM *Is in its Infancy in Europe.* San Jose (CA): Gartner Group, 2001.

[4] <http://www.cio.com/research/crm/edit/crmabc.html>

[5] Mark C.; Simon R., "*Sun Certified Enterprise Architect for J2EE™ Technology Study Guide*"

[6] Rainer A., Thomas P., "*Successful Practice in Customer Relationship Management*", In 2004, IEEE Proceedings of the 37th Hawali International Conference on System Sciences.

[7] Nykamp, M. *The Customer Differential: The Complete Guide to Implementing Customer Relationship Management*. New York: Amacom, 2001.

[8] Peppers, D., and Rogers, M. *One to One B2B: Customer Development Strategies For the Business-to-Business World. New York*: Currency, 2001.

[9] Treacy. M, & Wiersema, F., *Customer Intimacy and Other Value Disciplines, Harvard Business Review,* January - February 1993.

[10] <http://www.siebel.com/crm-company/reg-call.shtm>

[11] Chris Todman; *Designing a Data Warehouse: Supporting Customer Relationship Management. 2000*

[12] Maria P., Marcella. *The Reference Model Approach Successful CRM Implementation,* CRM Practice, ACQRA White Paper. 2002

[13] <http://www.ids-scheer.com/>

Web-Based Pornography – A Moral Challenge to an Information Technology Age

Longe Olumide Babatope; Longe, Folake, Adunni; Chete, Fidelis Chukwuweta

ABSTRACT

There is increased growing concern about the social and psychological impact obnoxious and spurious web contents will have on our society now and in the future. The invasion of pornographic sites on the Internet is an issue of serious consequences. It has been estimated that over 10 million people a week visit pornographic websites in Nigeria alone. The psychological effects of this and its attendant repercussions cannot be overemphasized. An increase in sex crimes resulting from these activities are already noticeable in some advanced countries of the world; this trend is expected to manifest itself in the third world. If something is not done and urgently too, the "global village" may as well metamorphose to a "global brothel". So far, the only panacea observed especially in Nigeria is the placement of notices in cyber cafes prohibiting the browsing of sex sites. More would have to be done technically. This paper x-rays the impact of Internet pornography on web-users in Nigeria and advocates the use of web filtering programmes as a robust measure against unwanted Internet content.

1.0 Introduction

The increasing impact and application of information technology expressed in the proliferation of computer networks worldwide, comes with both positive and dire consequences. In Nigeria, research has shown that over 35% of Internet access is related to the browsing of one sex site or the other (Longe, 2004). Other usage are distributed as follows; Entertainment (20%), Academic (15%), Sports and News (15%), Travel/ Business Information (10%) and Others (5%).

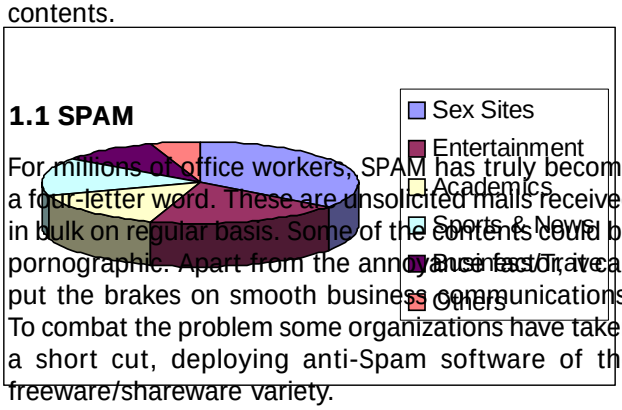


Fig.1: Pie Chart Showing the Distribution of Internet Usage in Nigeria

Apart from the popular data security and system protection issues, IT professionals, psychologists and the society at large daily express concern over a wide range of causes and effects associated with the use of the Internet for pornography and other immoral contents.

1.1 SPAM

For millions of office workers, SPAM has truly become a four-letter word. These are unsolicited mails received in bulk on regular basis. Some of the contents could be pornographic. Apart from the annoyance factor, it can put the brakes on smooth business communications. To combat the problem some organizations have taken a short cut, deploying anti-Spam software of the freeware/shareware variety.

Unfortunately these products tend to fall short user's expectations. Today's enterprise cannot afford the administrative overhead, inaccuracy and lack of support inherent with these products; they tend to trigger false hopes, often stopping business critical e-mail from reaching its destination (Edward, 2002). Our main concern in this paper however is another socio-cultural downturn, which has to do with the future of an upcoming generation that sits in front of the computers at home, in offices and cyber cafés around the world browsing sex and other immoral sites.

1.2 The Definition Of Pornography

In order to properly discuss pornography and be able to link it to moral decadence in the society, we must first come to a basic and agreeable understanding of what the word pornography means. The term pornography originates from two Greek words, porne, which means harlot, and graphein, which means to write. The combination of the two words was originally meant to describe, in literature, the sexual escapades of women deemed to be whores. As times had passed, this definition of pornography has grown to include any and all obscene literature and pictures (Edward, 2003). At the present date, the term is basically a blanket, which covers all types of material such as explicit literature (electronic or print), photography, films, and videotapes with varying degrees of sexual content (InfoSystems, 2003). The invasion of pornographic and other sex sites on the Internet is an issue of serious concern to the world at large.

1.3 Classification of Pornography

Catherine (2003) classified pornography into three distinct forms as follows:

(a) The sexually explicit and violent;

(b) The sexually explicit and nonviolent, but subordinating and dehumanizing; and

(c) The sexually explicit, nonviolent, and non-subordinating that is based upon mutuality (erotica)

Research consistently shows that harmful effects are associated with the first two, but that the third “erotica” is harmless. These three categories basically exist as tools of discerning content. Although sometimes they overlap without a true distinction as in when the content is graphic in the sexual act and also in violence, but shows the act as being a mutual activity between the people participating (Dewey, 2003).

2.0 The Anonymity Factor

The intention of the inventors of the Internet is to make information exchange faster, portable and accessible. The Internet in serving this purpose today comprised of over 6 million networked computers all over the world. As with every good innovation with an all comers entrant, certain unpalatable advantages is being taken of this facility by those who see it as a means not only to gratify fleshy ends, but also a means to legalise that which is morally illegal and cream off some money from those willing to patronize their abusive trade.

The Internet has popularized the sex business more than any other means of advertisement. This is aided by what psychologists called technology-induced anonymity. With unlimited access to a variety of websites, and the impediment of needing to enter a brothel physically removed, immoral gratification is just the click of a mouse away from any intending customer (Sackson, 1996). Anonymity has been an aid to most crimes perpetuated on the Internet and other IT applications. Pornography is not an exception. It can be viewed in the closet, on a laptop, on a palmtop etc without the reservation that any other person will know about the content being consumed.

2.1 Pornography and Consumers of Content

That which pornographic films, books and magazines have not been able to do from time immemorial, the Internet has achieved in less that a decade. The legal system in some advanced countries like the US and Canada contain laws that support pornography with clauses that it is meant for adults. These ideas will only get worse as society grows. Only through actions, discussions and proper education can the perpetuation of the negative impacts of pornography be swept from the closets and dark corners of the society (Awake, 2003).

For instance what means do we have for preventing those who are not adults; the youths and children from participating in such cankerworm since they are the most sexually active of every population strata and the ones addicted to web surfing? (Longe, 2004). Pornography has served as a means of unlocking evils buried inside for long; acting as a key to unlocking the evil in unstable minds. According to Edward (2003) the relationship between sexually violent images in the electronic and print media and the Internet in particular and subsequent aggression and callous attitudes towards women is much stronger statistically than the relationship between smoking and cancer.

After considering the increase in rape and molestation, sexual harassment and other sex crimes over the last few decades, and also the corresponding increase in the browsing of Internet sex sites, Internet pornography needs considerable study and examination. Substantiated evidence when evaluated and quantified show that habitual use of pornographic material promotes unrealistic and unattainable desires in men that lead to violent behaviour toward women (Longe & Longe, 2003). The need therefore arise to look for schemes that will assist in reducing if not eliminate the evil effects so far identified and associated with increased pornographic content on the Web.

2.2 Effects of Pornography

The effects of pornography can be summarized as follows:

- Women and young girls cannot have a healthy self-image of themselves when their husbands and fathers look at pornography. It is an evil addiction.
- It is dehumanizing
- It leads to increased incidence of rape
- It can lead to increased incidence of HIV and AIDS
- It usually comes with drug addiction
- It has led to an increase in home breakages
- It can create a chain reaction leading to violence and other moral & societal decay.
- Children are being sexually molested by those looking for sexual gratification
- Legal Threats**:** Employees can sue if organizations don't provide a work environment free of gender and minority harassment. That means taking reasonable care to block offensive Internet content.
- Network Threats: An employee can crash the computer network just by logging into the wrong website.
- Security Threats: Many viruses enter networks through web-based porno-content.
- Kidnapping is on the increase by those wanting to have it by all means with their website sex models.

3.0 Web Content Filtering: An Emerging Solution

It is a common fact that apart from the damage done psychologically to the life of consumers of content, pornographic sites are channels through which virus can invade servers and other systems on the network. So far, the only panacea observed in Nigeria is the placement of notices in cyber cafes prohibiting the browsing of sex sites. This is far from being effective as the browsing of sex site is on the increase daily. The level of prevention attainable by this measure is to say the least very low. The need therefore arise to employ technical measures that cannot be easily circumvented by some smart users (SieveNet, 2004).

3.1 The Internet Content rating Association

The Internet Content Rating Association is an international, independent organization that empowers the public, especially parents, to make informed decisions about electronic media by means of the open and objective labeling of content. ICRA's dual aims are to Protect children and youths from potentially harmful material. They also protect free speech on the Internet. There are two elements to the system. These are:

(a) Content Labeling: Web authors fill in an online questionnaire describing the content of their site, simply in terms of what is and isn't present. ICRA then generates a Content Label (a short piece of computer code), which the author adds to his/her site. Users, especially parents of young children, can then set their internet browser to allow or disallow access to web sites based on the objective information declared in the label and the subjective preferences of the user. ICRA's labeling system is designed to be as objective as possible, and to cover a wide range of content types. The system gives users a great deal of flexibility in their choices of what should and shouldn't be seen in their home or workplace. The browser's filtering system can of course be disabled and enabled easily. The broad topics covered are chats, the language used on the site, nudity, sexual content of a site and the violence depicted on the site and others such as gambling, drugs and alcohol. Within each broad category the web author is asked questions about whether a specific item or feature is present or absent on the site.

(b) Web Filtering Software: Web filtering programmes seems to be the most popular technical solution for now to the dangers of filthy web contents for private companies, in public libraries and in all computers used to access the Internet with a growing need to ensure that their computers are not being abused to access illegal or inappropriate materials. The use of valuable computer resources cannot be controlled unless a powerful set of monitoring and blocking tools are employed. Internet filters blocks SPAM e-mails and pornographic images transmitted to a web browser or via email by residing between a company's or ISP's (Internet service provider) connection to the Internet via both incoming and outgoing network connections. It examines the contents and images of a web site and can determine whether the site displays what is commonly accepted as pornographic content.

4.0 Advantages of Using Web Filters

The use of content filters helps prevent unwanted web content. They are able to identify words similar to those used in other healthy but related sites and thus does not block educational, medical, and/or sites that are political in nature-not intended to "arouse prurient interest"**.** They are cost effective and can be updated to combat new tactics developed by sex site developers. Improvement of existing techniques is a continuous research area to beat spammers at their game.

5.0 Conclusion

This paper has been able to x-ray the problems of Internet abuse in the light of increasing pornographic content on the web with special reference to the Nigerian situation. Experiences have shown that there is a determined effort among service providers; especially cyber-cafe operators to fight this menace but the unorthodox methods being used have not been effective. We therefore advocate the use of technical measures that have been proffered. Internet pornography is a menace that all IT practitioners will join hands with those providing the public with Internet access to combat. Our role in this regard cannot be overemphasised.

REFERENCES

Awake Magazine (2003): Pornography: Harmless or Harmful. Watchtower Publications, Benin, July, 2003

Catherine, I. (2003): Pornography Classified. Available online at <http://peacefire.ord/censorware>

Dewey, C. (2003): Internet Device Blocks SPAM, E-mail and Computer Viruses and other Suspected Contents. Available online at <http://www.sievenet.com>

Edward, D. (2003): Internet Filters that Blocks SPAM, E-mail and Pornographic Images. Available online at <http://www.sievenet.com>

Edward, B. (2002): Keeping Internet Predators at bay. Available online at <http://peacefire.ord/censorware>

Infosystems: Nigerian Tribune, Thursday March 6, 2003

Longe, F.A (2004): "The Design of An Information-Society Based Model For the Analysis of Risks and Stresses Associated Risks Associated with Information Technology Applications". Unpublished M.Tech. Thesis, FUT, Akure, Nigeria.

Longe, O.B & Longe, F.A (2003): Health Risks Associated With IT Applications: Issues and Solutions. Proceedings of the 7th International Conference of the Nigerian Computer Society. Abuja, Nigeria.

Sackson, M. (1996): Computer Ethics: Are Students Concerned. First Annual Ethics Conference. Available online at <http://www.maths.luc.edu/ethics96/papers/sackson.doc>

SieveNet (2004) Internet filter for Unwanted Web-Contents. Available online at http//www.net-sieve.com

Design, Development and Implementation of a Web enabled Home Devices Controller

Oluwaranti, A.I; Aderounmu, G. A; Adagunodo, E.R; Olateru, R.O.

ABSTRACT

This paper presents the implementation of a web enabled automation system for controlling home appliances over the World Wide Web (WWW). The system consists of three main modules, vis: the electronic device unit that handles the actual control of the devices, web based Graphical User Interface (GUI), and Apache web server. The electronic unit was implemented using FT 245BM microchip which is capable of controlling up to eight (8) devices via the USB port. The second module focused on the design of a web-based graphical user interface (GUI) using java programming language. Java was chosen because of its capability to handle web programming. Java applet, an aspect of Java programming was used in designing the basic HTML files while Macromedia Dream Weaver was used to put finishing touches to beautify the pages, which essentially deals with server-client connection. Apache web-server was used in achieving the client-server relationship in this work. In conclusion, this work was able to accomplish the task of changing the states (ON or OFF) of the connected devices over the Internet.

1.0 Introduction

Over the years, computers and computer communication had gained so much attention and had grown tremendously (Buchanan and Wilson, 2001). Technology has reduced the big and expensive computers of those days to portable and affordable devices. Most people nowadays can now easily afford computer systems at home. With the mass availability of personal computers, statistics have shown that people spend more time on computers than they do with friends and family members because most of activities that are needed to be done are either locally available on the computer or can be found on the Internet or over a network. <http://www.beyondlogic.org>

Since much time is spent on the computer system, it is therefore becoming a necessity for home appliances to be controlled via the computer system. This includes switching home appliances like electric fans, pumping machine, electric bulbs, television set, microwave oven, security lamps, and so on, on the computer system. This primarily will ensure full concentration on whatever activity is being performed on the system. In the same trend, the Internet has proven to be a good tool, by which several activities, which would have cost large resources and manpower to be invested in can now be carried out much easily and in a short time. The trend for some years now, has being e-commerce, e-education, and so on. The idea is to achieve on the Internet, what one would have done locally.

The Objective of this work is to control home appliances via a computer port on the computer system remotely over the Internet.

2.0 The concept of Home automation

The home automation concept has been part of the science fiction films for a while, where computer controls devices in the home and responds to voice activated controls. The Home automation & Networking Association defines home automation as follows:

"*.. a process or system (using different methods or equipment) which provides the ability to enhance one's lifestyle, and make a home more comfortable, safe and efficient...*"

From this definition, it shows that home automation can range from simply controlling of lights without leaving a seat, to being able to log onto the home system and switch on the TV and VCD to record a TV programme while not at home. According to Martin (2003) in order to achieve this concept of automation there has to be an interface between the home appliances and a central controller, the PC in this case. (Hastings, 2000). http://www.homeautomation.org

Generally, the term "interface' characterizes a hardware or software unit that establishes a connection between two or more different units. Initially, the origi-

nal concept of the PC was to provide users with their own complete computer systems, but with time, interfaces were required so that the PC can communicate with other devices. (Osinuga and Oresotu, 2001). There are two main methods of communicating with external devices; memory-mapped I/O or Isolated I/O. On most PCs nowadays, several interfaces can be found. Most of them have parallel and serial interfaces, USB ports, games port. The trend in PC technology is towards a standard interface unit, which is the USB port. <http://www.usb.org>

2.1 Universal Serial Bus (USB) Port

Universal Serial Bus (USB) is a connectivity specification developed by computer and telecommunication industry members for attaching peripherals to computers. USB is aimed at peripherals connected outside the computers. It eliminates the hassle to open computer case for installing cards needed for certain devices. "Universal" means all peripherals share the same connector (Messmer, 1997). Figures 1a & 1b depict a Type A & Type B USB Connector respectively.

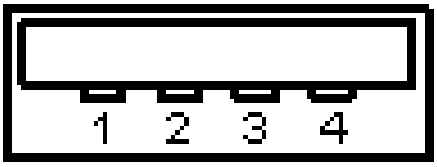


Figure 1a: Type A USB Connector

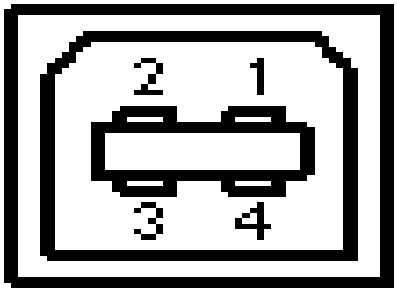


Figure 1b: Type B USB Connector

2.2 Web Server

A Web Server (also called HTTP Server) is a server software that uses HTTP to serve up HTML documents and any associated files and scripts when requested by a client, such as a Web browser. When the server gets a request from the client, it processes that request and returns some data (usually in the form of a formatted page with text and graphics). The connection between client and server is usually broken after the requested document or file has been served. HTTP servers are used on Web and Intranet sites. Basically, communication between Web servers and browsers or other clients is carried out using the HyperText Transfer Protocol (HTTP), - the client/server protocol used to access information on the World Wide Web. The markup language used for documents on the World Wide Web is HyperText Markup Language (HTML). Text formatted using HTML are arranged with *hyperlinks*-directly accessible connections-so that the contents of the document can be "read" in a non-linear fashion. By clicking on a hyperlink, the reader can jump around within the document, and even to other documents. Hypertext allows a user to refer to other documents stored in the same computer or in a computer located in a different part of the world. (Laurie and Laurie, 1999; Collins and Wall, 2002).

2.2.1 How Web Servers work

Whenever a Web server desires to get information from the server, it first performs a DNS lookup on the server name that is specified in the URL. The following are the sequence of steps that are followed when a client desires to communicate with server.

i. The Web client first performs a DNS lookup on the server name specified in the URL

ii. It obtains the IP address of the server and then connects to port (the default for http) at that IP address and a connection is established between the server and the client.

iii. The client sends an HTTP GET request for the document in the URL.

iv. The server receives the request and translates the document URL into a filename on the local system.

v. When the information has been fully sent, the connection between the server and the client is torn down.

2.2.2 The Apache Web Server

Apache was developed by a group of Web administrators who had earlier worked on NCSA (National Center for Supercomputing Applications) HTTP Server. The name Apache was taken from "A PAtCHy server" i.e. it is based on some existing code and a series of "patch

files". Apache implements the required functionality described in RFC2616, the document that defines the current Web server protocol (Laurie and Laurie, 1999). Apache runs under a suitable multitasking operating system. In Windows, its binary is called apache.exe, while in UNIX (and UNIX-related operating systems), it is called *httpd* (http daemon) (Horstmann & Cornell, 2000).

3.0 Design Methodology

In the previous section, the concept of home automation was introduced and other related issues were discussed. This section provides a description of the component used in the design and actual design methodology.

3.1 Description of Components

Several microchip components were used in this design, the essential ones are described below:

a. FT245BM

This component serves as an interface between the device and the computer. The FT245BM is a chip used in connecting to a host PC via USB. It is a chip that provides a method of transferring data to/from a peripheral and a host P.C. at up to 8 Million bits (1 Megabyte) per second. It is designed in such a way that it interfaces the CPU either by mapping the device into the Memory / IO map of the CPU, using DMA or controlling the device via IO ports.

When the host PC sends data to the peripheral over USB, the device will assert the receiver full status bit to let the peripheral know that data is available. The peripheral then reads the data until the receiver full status bit goes inactive, indicating no more data is available to read. They also offer "hot swapping", are self-powered (up to 500 milliamps). The electrical interface between the microcontroller and the FT245BM is comprised of eight data lines and four handshaking lines.

Figures 2 and 3 show the FT245BM IC pin out and its simplified block diagram.

b. Opto-coupler - MCT2E

Opto-coupler (also called opto-isolator) is a LED emitter combined with a photodetector. Opto-couplers are light-coupled isolators which allow digital (and sometimes, analogue) signals to be set between circuits with separate grounds. They provide 2500 volts (rms) isolation, 10^{12}Ù insulation resistance, and less than a picofarad coupling between input and output. MCT2E is the opto-coupler used in this work. Figure 4 shows the pin layout of MCT2E.

Specifications

Input diode	
Forward dc current	60mA
Reverse dc voltage	3V
Peak forward current	3A
Power dissipation	100mW
Total Power Dissipation	250mW
Output Transistor	
Collector-Emitter voltage	30V
Emitter-Collector voltage	7V
Collector-Base voltage	7V
Power dissipation	150mW

c. Transistor - 2N2222

A transistor is basically an active device that can amplify, producing an output signal with more power in it than the input signal. This additional power comes from an external source of power. (Horowitz and Hill, 1989)

The Transistor 2N2222 is an npn type switching transistor with the following characteristics:

i. Collector is more positive than emitter.

ii. Base-emitter and Base-collector circuits behave like diodes. i.e. Base-emitter diode is conducting while the base-collector diode is reverse-biased, in other words, the applied voltage is in the opposite direction to easy current flows. Figure 5 shows the schematic diagram of the transistor.

4.0 Design Circuits Description and Implementation

The design objective of this work can be represented diagrammatically in figure 6. The connection of the hardware components used for the electronic design is described in figure 7. The major component used in the design is FTDI-FT245BM, which was described in section 2. Pins 17 through 24 of the component serves as the data inputs for each of the devices. Pins 4, 10, 11 and 12 are connected to the pnp MOSFET transistor IRLML6402CT which serves as the current breaker that prevents the components from current overloading.

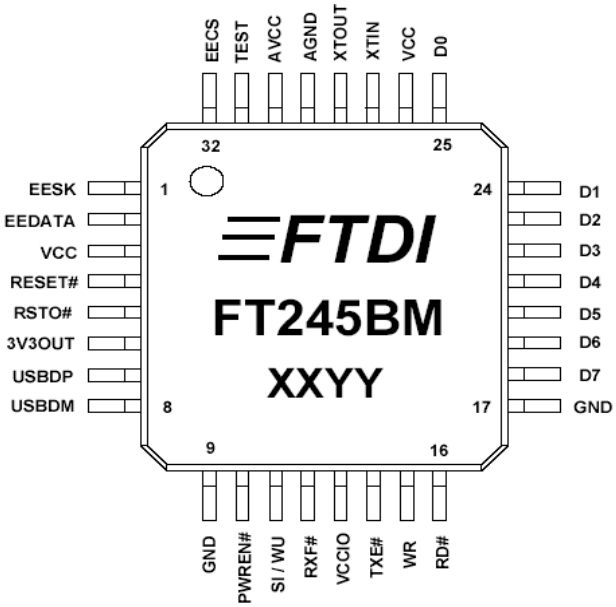


Figure 2: FT245BM Pinout

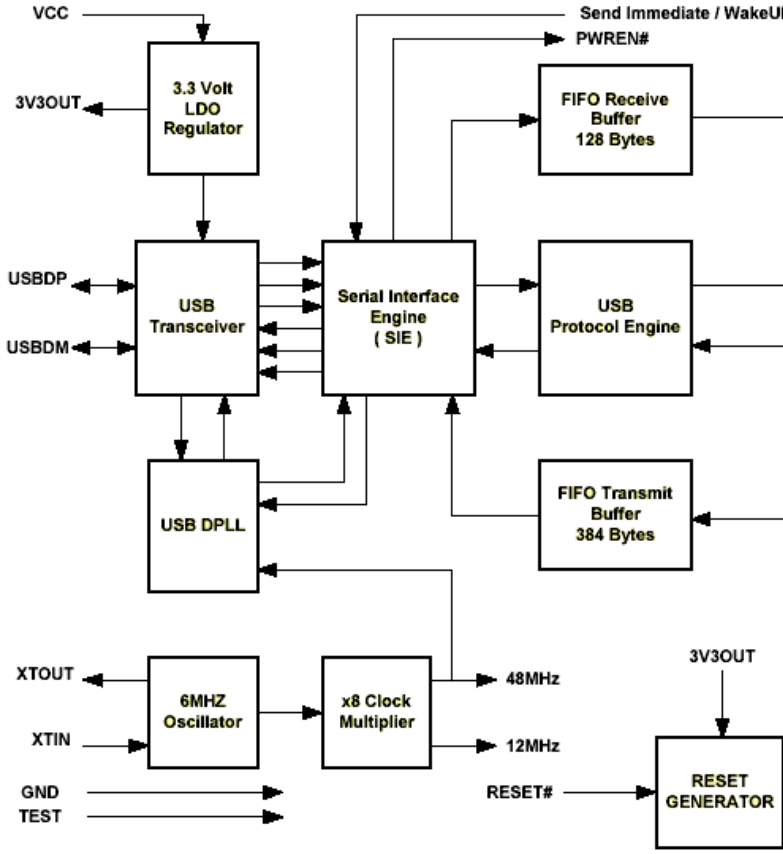


Figure 3: FT245BM Simplified Block diagram

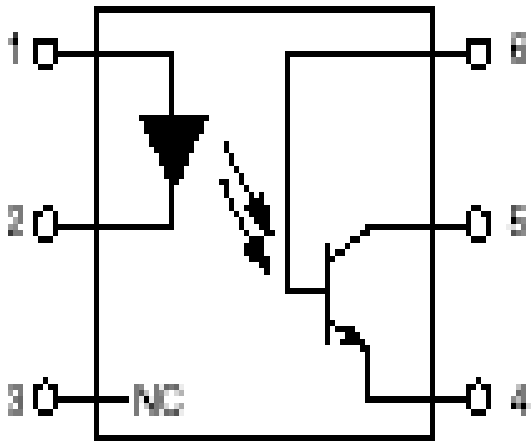


Figure 4 Pin layout of MCT2E

Figure 5 Schematic diagram of 2N2222

4.1 Description of Graphical User Interface

Since the objective is to be able to control (switch ON and/or OFF) home appliances. Therefore, for data to be properly transferred via the Internet, they must be described in HTML format. For this reason, Java Applet was used in designing the basic HTML files while Macromedia Dream Weaver was used to put finishing touches to beautify the pages. Figures 8, 9 & 10 show the homepage, current status page and distinct device page respectively.

4.2 Java Applets and Security Issues encountered

Java applets were intended to be used in accomplishing the task of controlling the devices via a browser in this work. For this to be accomplished a program was written in Visual Basic 6.0 and the program was then made as an executable file. The idea is that this executable file be loaded when an action takes place. This worked well as an application, but did not, either by just running it using the CLI (Command Line Interface) or in browser. It was discovered that for the applet to run without generating errors, a policy needs to be set so as to permit the applet to run in an applet window. Due to some security issues encountered when attempting to write the codes in C language and then incorporating it in Java Native Interface - (JNI), the policy was created to allow the *Runtime.getRuntime()* object in *java.lang.RuntimePermission* class. The policy was named *all.policy*. Therefore to run an applet (say First.html that runs a compiled code of First.java) to be viewed in an applet window, the following command is required:

appletviewer -J"-D.java.security.policy=all.policy" First.html

Another problem encountered was when the applet was to run in a browser. As a standard, applet security in browsers strives to prevent "untrusted" applets from performing potentially dangerous operations, while simultaneously allowing optimal access to trusted applets, therefore, before granting optimal access to applets, adequate care must be taken to ensure security.

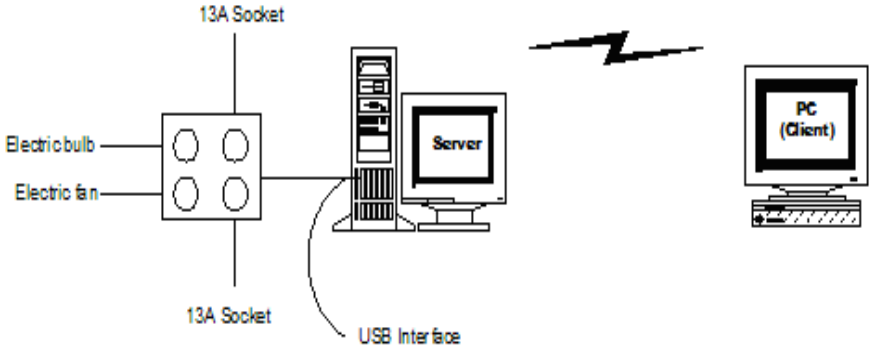


Figure 6: Schematic overview of the system

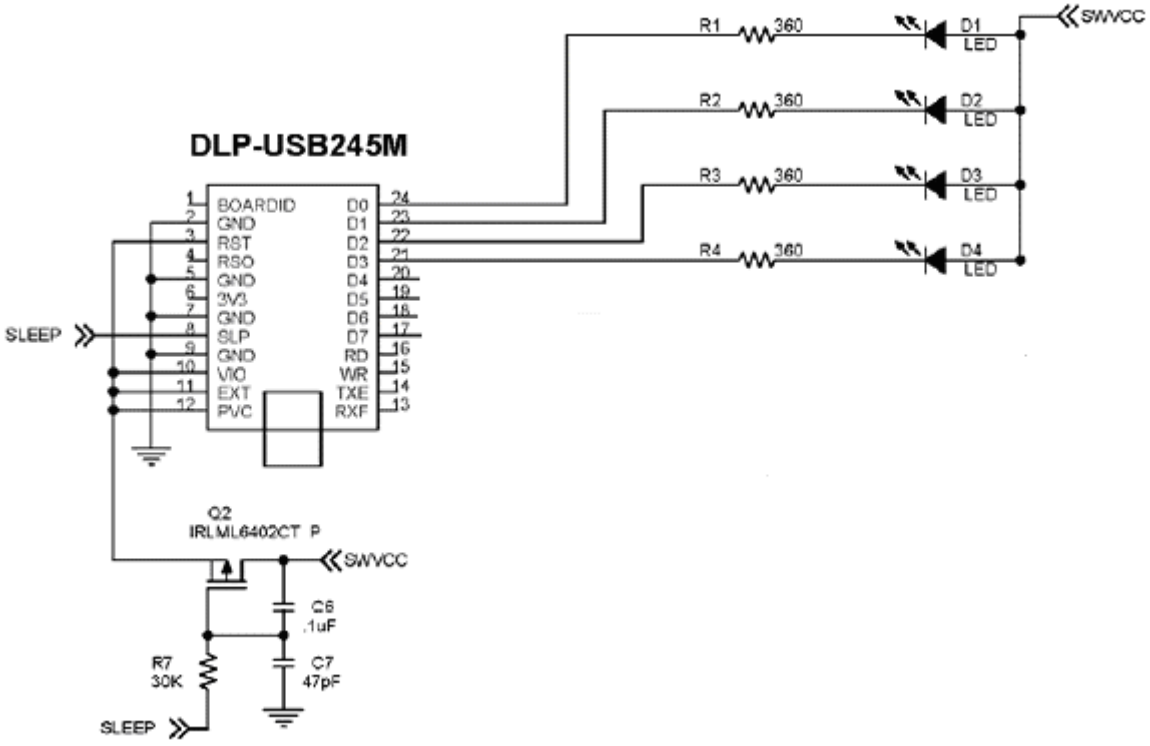


Figure 7: Schematic Circuit diagram of the Electronic design using the USB port

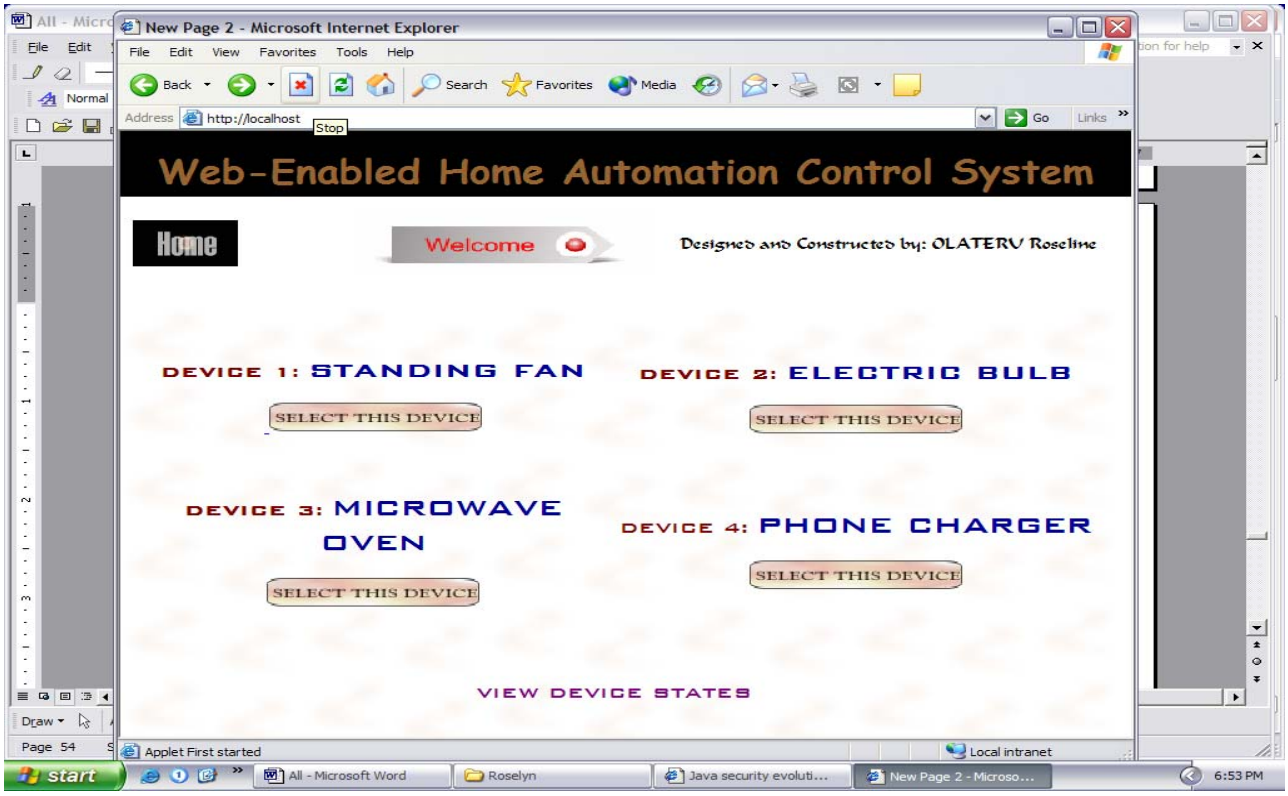


Figure 8: Homepage of the work

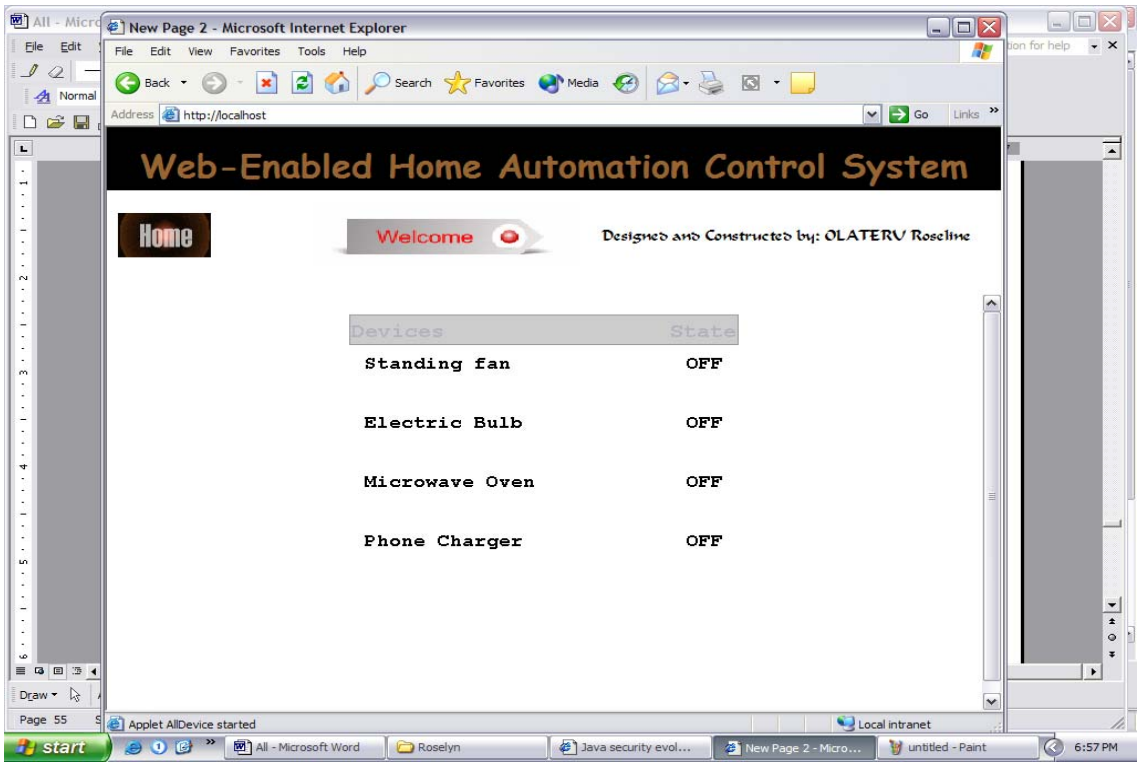


Figure 9: Page showing the current states of all the devices

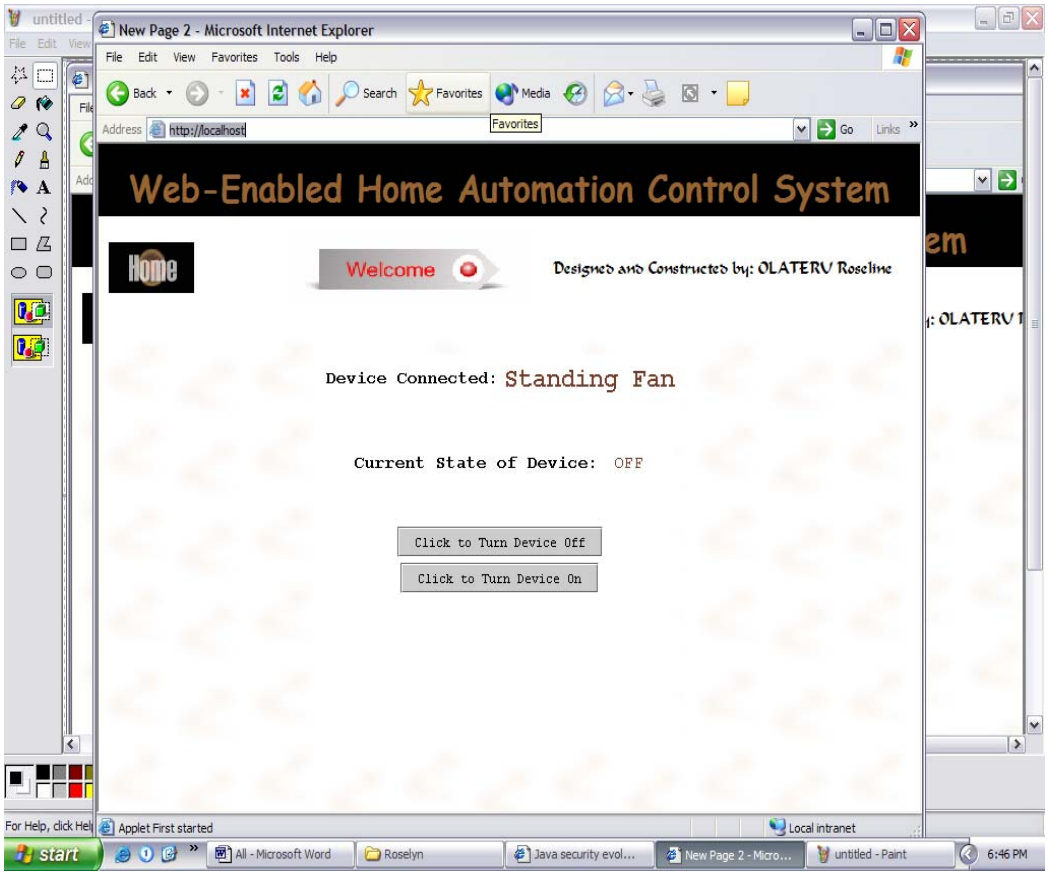


Figure 10: Page for a distinct device

The Java plug-in supports the standard Java 2 SDK, Standard Edition (J2SE), including the security model. All applets run under the standard applet security manager, which prevents potentially malicious applets from performing dangerous operations, such as reading local files. RSA-signed applets can be deployed using the Java plug-in. RSA signatures (which can be acquired from a Certifying Authority - CA) require that a few hundreds dollars be paid. Another alternative is to use a self-signed signature which can only be done in other operating systems.

Therefore, for this work, a socket program was written which consists of two programs - *Server.java* and *Client.java*. After compilation, the Server program is first loaded which permits the server to listen at a particular port, the client program is then loaded which connects to the server. Once ***on*** is typed at the client side, the device will be switched on, also to turn the device off, ***off*** is typed at the client's side. To suspend the connection, it is required to type TERMINATE.

5.0 Conclusion

Time, safety and security are three important factors. It is well known that time lost can never be regained. This work has been able to use the advantages of the Internet (or any existing network) to control home devices which invariably save time it takes to physically go from one location and the other to ensure that proper measures are put in place. It also means that one can turn on the boiler few miles from home.

Safety is also an important issue to reckon with; so many resources have lost to various fire hazards whose root had been traced to voltage instability of home devices. This work can help to prevent hazards of this nature by remotely controlling the home devices. Of importance also is security, being able to turn on security light at home while far away in the office is not just fun, but also for security reasons.

The system was demonstrated to control an electric standing fan but it can also be practically implemented to control just any home device.

REFERENCES

Buchanan, W and Wilson, A. (2001) *Advanced PC Architecture*. 1st Edition. Addison Wesley Press, England.

Collins, T. and Wall, K. (2002). *Hungry Minds Red Hat Linux Networking and System Administration.* 1st Edition. Hungry Minds Inc., New York.

Hastings, W. (2000) *Auto-MATE: Intelligent Home Automation Using Mains Power Communications*. Final Year dissertation. B.Eng. thesis. University of Queensland.

Horowitz, P. and Hill, W. (1989) *Art of Electronics*. 2nd Edition. Cambridge University Press, Massachusetts.

Horstmann C. S. and Cornell, G. (2000) *Core Java™ 2: Volume I-Fundamentals*. 5th Edition. Prentice Hall PTR, USA.

Laurie, B and Laurie, P (1999). *Apache: The Definitive Guide*, 2nd Edition. O'Reilly & Associates Inc, Canada.

Martin, A. (2003). *Investigating Home Automation*. Final Year dissertation. B.Sc. thesis. Sheffield University. <http://www.project.con-fusion.net>

Messmer, H. (1997). *The Indispensable PC Hardware Book*, 3rd Edition. Addison Wesley Press, England.

Osinuga M.F. and Oretsotu O.M. (2001), *The Design and Construction of a Computer Interface developed for the control of lighting and Home Appliance*. 'Unpublished B.Sc. thesis', Obafemi Awolowo University, Ile-Ife, Nigeria.

http://www.homeautomation.org/index_ie.html

http://www.beyondlogic.org <http://www.beyondlogic.org>

http://www.cert.org <http://www.cert.org>

http://www.javaworld.com <http://www.javaworld.com>

http://www.ftdichip.com <http://www.ftdichip/com>

http://www.usb.org <http://www.usb.org>

<http://www.senet.au>

Complementary Approach of Schema Evolution in GIS Databases

Adewole A. Philip

ABSTRACT

This paper addresses database development and schema evolution from the geographic information system integrator's point of view. The applicability of incremental approach to database schema evolution is considered, and incremental data evolution is proposed as a complementary method for schema evolution and data restructuring of large geo-spatial datasets in operational use.

1. INTRODUCTION

Designing models that evolve over time is still a major problem today in information technology (IT) industry. GIS is not an exception. While user requirements change quite frequently over time, databases continue to show little flexibility in supporting these changes in their structures and data organization. The main reason for change is that the perception of the real world is continuously evolving, or the part of the real world is that is of interest keeps on expanding because the need for new user or application requirements has arisen. Finally, there are also technological reasons calling for database evolution, such as changes in the functionality of the supporting computer infrastructure or internal system reorganization tailored for improvement in performance.

According to European Union, geographic information in Europe is about 38.5 per cent of all public sector information [European Union, 2003]. The computer penetration into the field of GI follows a general pattern: First it is adopted in the GI provider organization for speeding up the production process. In the second phase the potential of computerized environment is exploited more widely and the market for GI in digital form starts to develop In the third phase completely new products and activities arise from the digital culture.

The pioneer organizations that started to digitize GI early, have faced all the stages one-by-one. However, the stages are concurrent and ongoing, and are shown as layers of activities in Fig.1. In the time being the 'geo-e-business' will be absorbed into the 'normal' service production layer, but at the moment it is feasible to examine it as a layer of its own. The project portfolios of organizations tend to grow into new topics, and the most appealing questions from the manager's or technologically oriented researcher's point-of-view may lie in the expansion of technology and new applications, rather than re-engineering the existing parts. However, database (DB) and data management are the true cornerstones of the whole and they take effect on each activity layer, directly or at least indirectly. Commercial solutions for geographic data management (GDM) have developed rapidly during the past few years, showing the evolution of architecture from rather closed monolithic systems towards client-server architectures with open standardized interfaces. GI organizations have made great investments to establish their existing large datasets in digital form. Their concern for their asset is here-fold. Firstly, how to maximize the exploitation of the digital content. This is the driving factor for expanding into new types of products and services. These actions are very apparent and promoted. Secondly, the method and techniques in the data production layer need improving for more economic and efficient data updating. Last but not the least, user organizations are concerned as to how to keep up with the evolving data management technology and ensure the sustainable platforms for their information asset. Because GI datasets are typically *data-intense* both in size and complex structure, the solutions and new arrangements in data management have significant impacts on all actions. For user organizations of GIS, database re-design and schema evolution are challenging tasks in the development of their next-generation systems.

This paper focuses on : databases in next-generation system project, taxonomy of database development activities, factors of database development activities, and the incremental data evolution approach to DB schema evolution.

2. DATABASES IN NEXT-GENERATION SYSTEM PROJECTS

Next-generation GIS projects are current in many organizations. Next-generation GIS projects usually include a database project, where data is translated

from the existing data storage into a new database management system (DBMS). A DBMS with spatial capabilities has been strongly associated to certain vendor specific client solutions and product families, but the situation is changing rapidly. Currently one has alternative clients, components and development tools to choose from. It is also possible to implement the 'business intelligence', or the 'geo-spatial intelligence' in different layers of the software architecture: the database server, the middleware or the client.

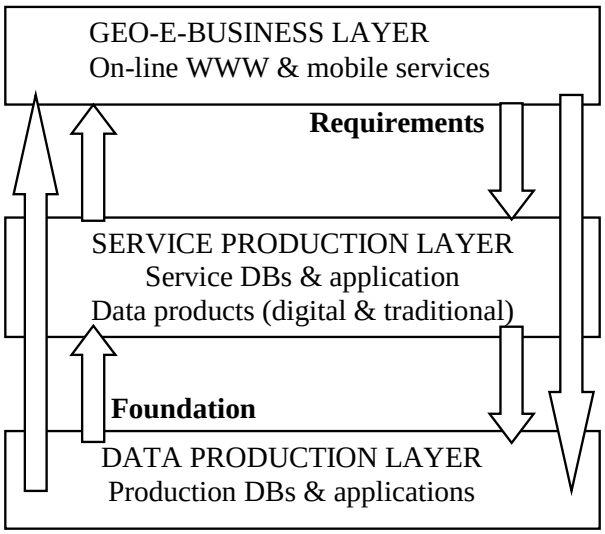


Figure1. Activity layers of a GI provider organization

In a next-generation GIS project, the user organization needs a practical solution for database management. Although the latest technology always seems to be three months ahead, one just has to try to make the best out of what is available here and now (Kucera et al., 2002). According to [Salo-Merta et al., Salo, 2004] a practical solution was defined with three keywords: Economy, Maintainability and Technical requirements. *Economy* stands for both investments cost and operational costs. Maintainability stands for the wish to choose a provider and platform with a good probability to survive and be competitive and technology advanced in the future. *Technical requirements* include general and specific database capabilities, and issue concerning architecture, interfaces, scalability, extendibility and programming facilities. All of these need to be examined in the context of the user organization, to determine the 'customer need'. As the DBMS is a central component of the system, major development actions involve needs to develop the other subsystems as well. In the most dramatic case it means the replacement of existing clients with new ones.

3. TAXONOMY OF DATABASE DEVELOPMENT ACTIVITIES

The development of a system's DBMS environment is based on three basic types of activities: *Database product change, Schema Versioning and Schema evolution* (Fig. 2).

3.1 Database Product Change

The database product change projects include:

- Design and implementation of the new DB schema
- Preparation of data translation
- Data translation
- Data validation
- Further data re-structuring

It is noteworthy that because of the history of proprietary DBMS solutions for geographical data, considerable effort may be required for other software activities in database product change projects. It may be necessary to change the client applications as well or to re-write the interfaces to the existing ones. For this reason the activities concerning the database itself are probably only a part of a large software project, and the pressure to minimize the size of the project by altering the data as little as possible at this stage may be substantial.

3.2 Schema Versioning

A database supports schema versioning if different versions of its schema can be identified and selected by a suitable labeling system, such as symbolic naming, or time stamping. Schema versioning can be partial – in which case updating data is only allowed on one specific schema – usually the current one, and the old data recorded with a previous schema can be considered "read only". Full schema versioning supports viewing and update of all data, through a user definable versioned interface. A modification of a schema does not necessarily result in a new version, and often may be of finer grain.

Schema versioning may be "explicit", in which case the system expects the user to have knowledge of the schema of the database, and hence to use the appropriate access software. Implicit schema versioning is more user friendly, but of course more difficult to implement. In this case, the user is expected to have knowledge of the semantics of the managed data, which may or may not be a real database schema, and the user interface will have the capability to direct

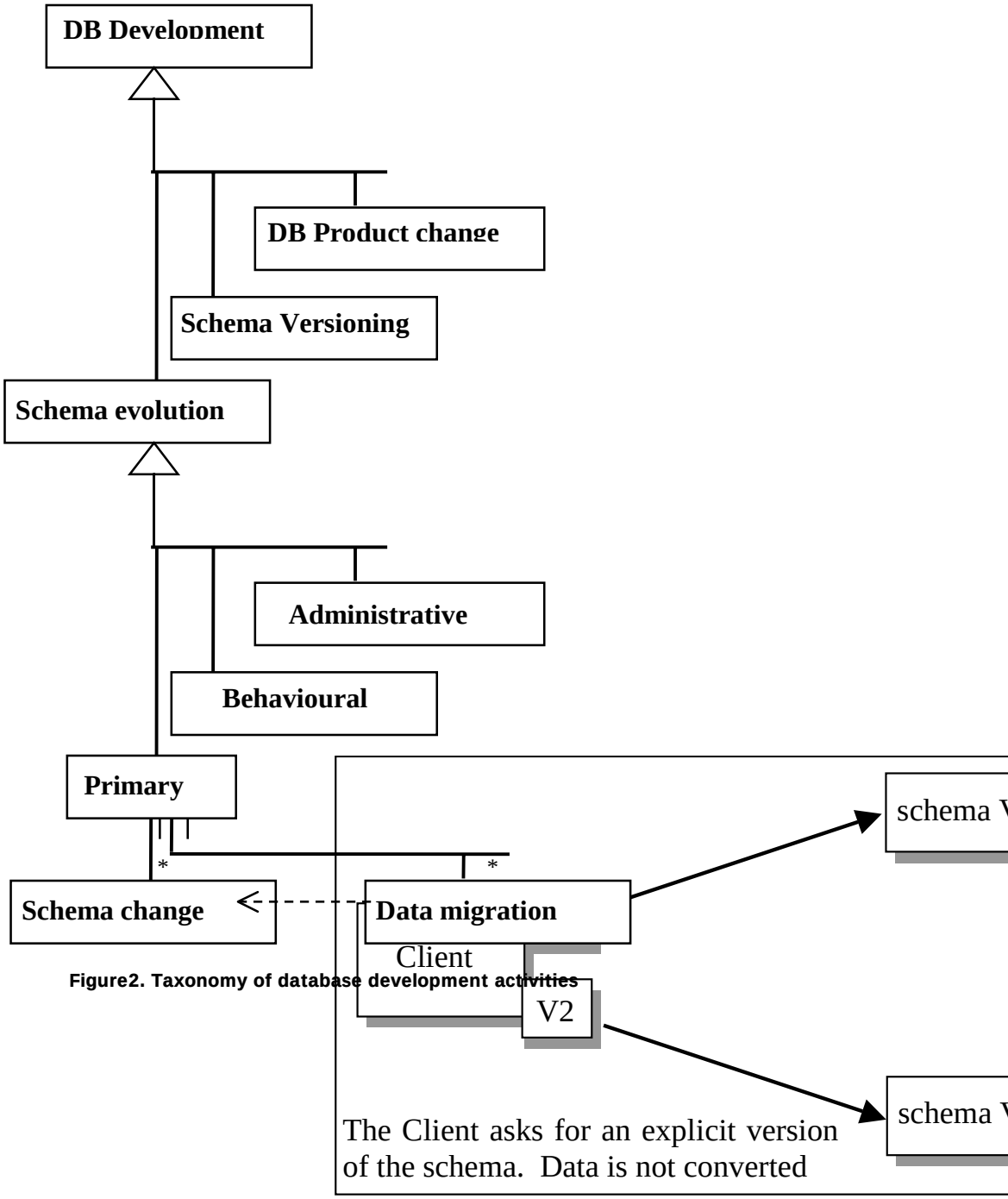


Figure2. Taxonomy of database development activities

Figure 3: Explicit Schema Versioning

database operation to the correct version of the schema and to a coherent set of data stored in the database. Such a system can be described as a "schema independent".

3.3 Schema Evolution

A database supports schema evolution if a modification to the schema does not result in the loss of any existing data. This means that the software must be backward compatible. New versions of the software must access old data, acquired using an old schema. Hence backward compatibility implies that data will be converted from the old schema to the new schema. More formally, all valid instances of a schema S_0 must also be valid instances of a later schema S_1.

4. FACTORS OF DATABASE DEVELOPMENT ACTIVITIES

The factors that influence database development activities are discussed below. They are: Data conversion cost. Evolution strategy, Developing DBs in operational use and Pluralism in DBs.

4.1 Data Conversion Cost

Converting the data into the new system as quickly and painlessly as possible may be a key question in an organisation's DBMS implementation project. Therefore, commitment to the old design may become a critical factor in the design of the application data model. [Salo-Merta et all., 2004]. For a very large database with objects with complex structure, which geographic objects appear to be, each extra step included in the conversion process may have a dramatic effect in the throughout time, if it causes a need for operator intervention. Data quality may become the bottleneck in the conversion process, if the target system's validation rules are tighter than the source system's.

4.2 Evolution Strategy

All the user requirements can seldom be satisfied at once, and they keep on changing. Yet the data acquisition for digital geo-spatial datasets is expensive and time consuming, there is a strong motivation to preserve the existing data and pass it on to the next system generation. The lifetime of a dataset can exceed the lifetime of the DBMS that is used to maintain it (Salo-Merta et al., 2004). Therefore the development of the data system's database environment gets a step-by-step evolutionary nature.

In a step-by-step evolution one proceeds with a series of development steps to achieve the desired state. For example, in the Next-Generation System Project of the Topographic Data System, the primary data loading into the new database included basic validation of attributes and topological conditions, and polygon reconstruction. This case contained the merging of objects that had previously been fragmented by map sheet's edges. (Salo-Merta et al., 2004).

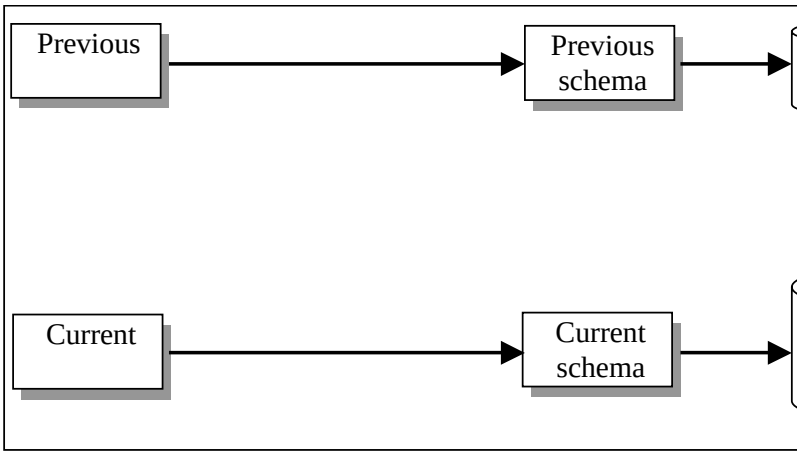


Figure 4: Schema Evolution

constraint in the system development and design, because the operational use of the database should not stop during a lengthy conversion phase. It may not be possible to stop the production for a full conversion. Instead, other alternatives have to be considered.

4.3 Pluralism In Databases

Kucera (1999) discussed pluralism in geospatial databases and suggested the acceptance of data heterogeneity in contrast to a monolithic database, which reconciles any pluralism as part of the update process. In a pluralistic system a geographic feature may have several different representations stored 'as-is', as they existed in provided datasets, without attempts to integrate schemas, generalize, resolve spatial discontinuity, or otherwise encourage consistency in representation. Pluralistic data management and representation require specific techniques: e.g. versioning, metacontent description, feature linking with same as links, on-line schema mapping etc.

5. INCREMENTAL DATA EVOLUTION APPROACH

5.1 Origin Of Incremental Approach

The incremental approach originates from the software industry, where it has been applied for incremental development and compiling of software modules. In the context of GIS, it has previously been proposed for genelisation [Kilpelainen & Sarjakoski, 2001]. Incremental updating of database has recently become an issue of research and development. The International Cartographic Association established a working group on Incremental Updating and Versioning in 1999. The working group's main interest relies on the management of updates between the base dataset supplier and the value-added dataset provider. Research issues include: bi-directional, multi-level, historical and temporal updating, planning for future changes, database maintenance, feature identifiers, modularity (dimension, context, layer, theme an size), inconsistent updating and simultaneous updating by field teams [Copper & Peled, 2001]. The implementation of new features to support incremental updating and versioning are likely to put pressure on product change or schema evolution on existing systems as well.

5.2 Production Discontinuity Problem

Traditionally database schema evolution is normally being carried out by *full data conversion strategy* using the following sequence:

- Stop Production
- Change the schema/Create a new schema
- Migrate all data to the new structure defined in the changed schema
- Continue production with the new schema

Here it is assumed that schema evolution includes such structural changes to the data that a specific software module is required for migration (Fig. 5). This migration module may be a SQL- script, a database program, an application program or a translation process with external transfer files. The characteristic of the migration process is the discontinuity that it causes to normal production. Normal production is carried out with software A using DB "interface a" .Data that conforms to the new schema, has to be accessed with an updated interface a'. Therefore two versions of the DB interface are needed.

This approach can be described as one very-long transaction. From the production applications' point of view, the database is in inconsistent state during the migration. Since the migration starts, the whole database is not accessible for client A 1 anymore, but on the other hand, only the migrated objects are accessible to client A2.

If the basic problem is the duration of data change and it is assumed that it cannot be solved reasonably, ways of managing the production and the migration simultaneously should be considered. In some data systems feasible partitioning criteria can be found, e.g. in working area based map production, but this is not always the case.

5.3 Principle Of Incremental Data Evolution

It may be a characteristic to a data system that the updates are made objects-based and on-request. The requests scatter all around the dataset and no partitioning criteria for isolation can be found. In that case the acceptance of pluralism in schema definitions of object classes and means to manage it are suggested. An approach called *incremental data evolution* (IDE) is proposed. Incremental data evolution is an approach where only those objects that are touched as part of the normal production process are migrated to the new structure defined in the changed schema.

Although a feature may have alternative schema definitions as a permanent or at least long lasting state

Figure 5: Migration module propagates schema changes to data. Migration causes discontinuity in production, because application program's DB interface needs updating to conform the new schema.

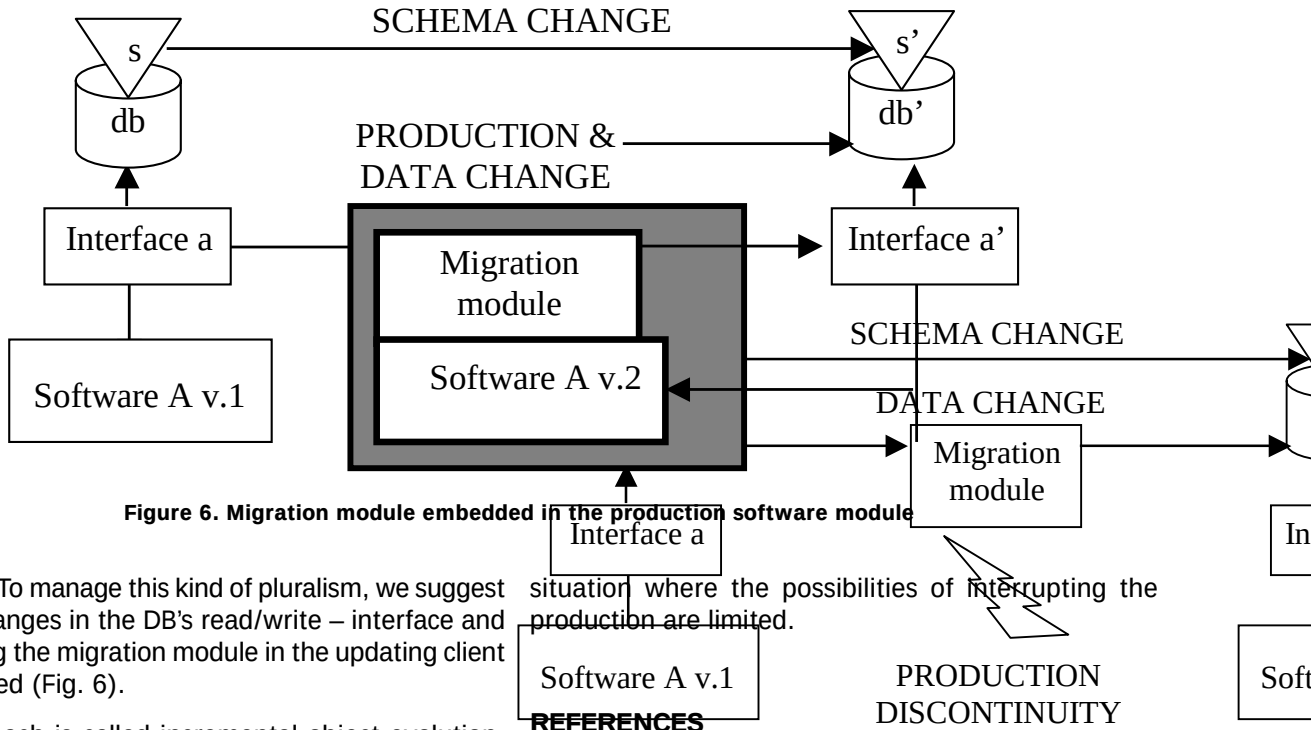


Figure 6. Migration module embedded in the production software module

of the DB. To manage this kind of pluralism, we suggest certain changes in the DB's read/write – interface and embedding the migration module in the updating client is suggested (Fig. 6).

This approach is called incremental object evolution, because the schema evolution is propagated to the data incrementally, as the objects are touched in the normal production. It is not forced to the whole dataset, but only to those objects that are most actual.

6.0 CONCLUSION

This paper considers the applicability of incremental approach to database schema evolution, and proposes a complementary method for schema evolution and data re-structuring of large GI databases. It is proposed and considered as a resolution to a practical problem of schema evolution of a production database, in a situation where the possibilities of interrupting the production are limited.

REFERENCES

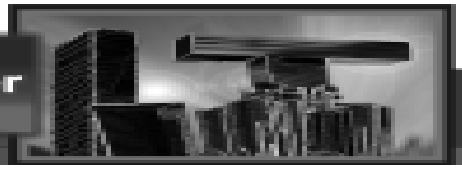
BaBar, (1999). Schema Evolution Strategy for the Conditions Database, BaBar Databases Group.

Cooper, A., Peled, A., (2001). Incremental updating and versioning. In: *20th International Cartographic Conference, Conference Proceedings,* Beijing, China, Vol. 4, pp.2801-2809.

European Union http://www.cordis.hi/euroabstracts/en/june03/feature02.htm

John F. R., (2003). A survey of schema versioning issues for database systems, Information and Software Technology, 37(7) pp.383-393.

Kilpelainen, T., Sarjakoski T., (2001). Incremental Generalization for Multiple Representations of Geographic Objects. In: Muller, J-C, Langrane J-P, weibel R., (eds) GIS and generalization, Gisdata I, ESF, Masser, L., Slge, f. (series eds.), Taylor & Francis, pp. 209-218.



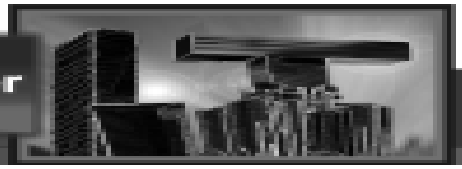
Kucera, G., (1999). Pluralism in spatial information systems. Paper on the IV International Conference on GeoComputation, USA on 25-28 July 1999.

http://www.geovista.psu.edu/sites/gencomp99/Gc99/060/gc_06

Kucera, H., Lalonde, B., Lafond, P. (2002). GEO 2002: Designing and Building a Spatial Information Warehouse, *19th International Cartographic Conference, Ottawa, Canada*

Luc G., (2002). Schema Evolution in Atlas.

Salo-Merta, L., Tella, A., Vanhamaa, 1. (2004). Database Design in Migration from Traditional to Object-Oriented GIS – the Evolution Story of the Topographic Database of Finland. In: *20th International Cartographic Conference, Conference Proceedings,* Beijing, China, Vol. 2, pp. 1393-1400.



Application of Geographical Information Systems to Educational Planning

Martins Fabunmi; Omotayo C. Dalumo

ABSTRACT

This paper examines the extent to which geographical information systems (GIS) can be used while planning education. In Nigeria, the manual method is used to plan education. But certain education data can not be processed manually because of its large volume and complexity. The geographic nature of education data makes the computer-based management information systems (MIS) to be unsuitable for processing such data. Hence, the need to adopt GIS technology to educational usage. GIS has the additional capability of storing, processing, manipulating and displaying geographic information. This study has implications for the management and also for policy makers. GIS technology is recommend for planning of education. Appropriate policies should be initiated to accommodate this recommendation.

Introduction

In educational planning, GIS is very essential for analysis, because of the geographic dimension to education. Education is a public service that is provided over space or in varying geographical areas and conditions. Hence, the need for all forms of geographical analyses. Most of which involve diverse and large volumes of geographic data sets. These data may be in form of maps, tables of secondary data, primary data, list of names and addresses. Manual methods cannot be used to handle such data. Neither are non-GIS computer systems suitable for processing such geographic data. However, when such data are input into a GIS, they can be easily manipulated and analysed in ways that would be too costly, too time-consuming, or practically impossible to do without using GIS technology.

Conceptual Framework

Education has a geographic dimension. Hence, geographical information systems (GIS) are suitable for its planning. Planning in the context of this paper means a process of preparing a set of decisions for action in the future with the view of achieving goals, by optimal means, while educational planning refers to the continuous process of obtaining and analyzing facts, and from empirical base, of providing information for decision makers on how best the education system is to accomplish its goals and on how to achieve cost-effectiveness of education programmes (Fabunmi, 2004). Geographical information systems are computer-based systems which are used to store, process, manipulate and display geographic information (Aronof, 1989 and Eastman, 1996). A GIS is designed for the collection, storage and analysis of objects and phenomena where geographical location is an essential characteristic or critical to the analysis.

The essence of GIS application in educational planning is to enable automated storage, processing, manipulation, display and dissemination of spatially referenced educational data. Eastman (1996) argued that a typical GIS comprise different components. It is not all systems that have all of these elements. But a true GIS package must have an essential group.

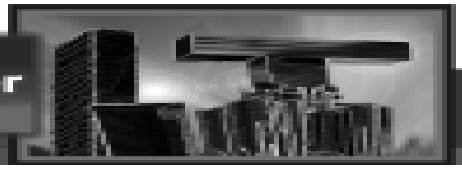
The essential components of a GIS include:

(i) **Spatial and attribute database**, that is, a collection of maps and associated information in digital form. For example, a school land can be defined in the spatial database as its qualities, such as its land use, land are, owner, property valuation, excetra, can be in the attribute database.

(ii) **Cartographic display system,** which allows one access to certain hard copy devices like printer or plotter. For instance, IDRISI for Windows allows for highly interactive and flexible on-screen cartographic composition which can be saved for display, or even printed.

(iii) **Map digitizing system**, which enables them into digital form. Thus further developing the database.

(iv) **Database Management System** (DBMS), which refers to a type of software that is used



to input, manage and analyze attribute data.

(v) **Geographic analysis system**, which has the ability to analyze data based on truly spatial characteristics like geographic location. For example, we may want to know the number of schools in urban centres, rural areas, hilly regions, riverine areas, excetra.

(vi) **Image processing system**, which has the additional capability of analyzing remotely sensed images and providing specialized statistical analyses. IDRISI for Windows is also suitable for this task.

(vii) **Statistical analysis system,** which process both the traditional and spatial statistics.

(viii) **Decision support system**, which assists the management in rational decision making. It is very useful while using multi-criteria to allocate resources. It is a powerful tool for decision-makers.

Operational Areas in Educational Planning

Educational planners perform a lot of operations with the view to ensure that educational facilities and services are equitably distributed. Most governments lay emphasis on the issue of equity in education. In Nigeria, both the 1998 Revised Edition of the National Policy on Education and the 1999 Constitution guarantees equal access to education. Hence, educational planners have to perform certain operations and monitor the education system before compliance with the laws can be effected.

Fabunmi (2004) listed the following as the operational areas of educational planners:

- Planning of alternative decisions for policy makers and executors, that is, administrators.
- Setting goals and objectives for educational systems.
- Planning of educational programmes and services.
- Planning the manpower needs of the country, state, local government area or institutions.
- Financial planning, with a view to provide guidelines on how to make prudent use of available financial resources.
- Planning the judicious utilization of available physical resources or facilities.
- Planning education within the existing governmental structure so that the education plan will enjoy the required governmental support for implementation.
- Planning education within the existing social context so that the education plan will enjoy popular support required for successful implementation.

In order to accomplish these tasks, educational planners often prepare

educational policies, initiate education plans, cost education plans, prepare school maps, diagnose educational situations, analyse education systems, perform demographic analysis, use statistical and projection models to illustrate the past, present and future of the education system. All these tasks look simple. But they are difficult to perform manually. Hence, the need to apply computer-based management information systems.

Fabunmi (1998) recommended the coupling of Management Information Systems (MIS) and Geographical Information Systems (GIS) packages for the performance of certain educational planning tasks, such as school mapping exercises. MIS is a suitable tool for planning operations that do not involve the use of spatially referenced data. The nature and purpose of education necessitate planning with spatially referenced data. Hence, the need to couple GIS and MIS tools for effective performance of educational planning tasks.

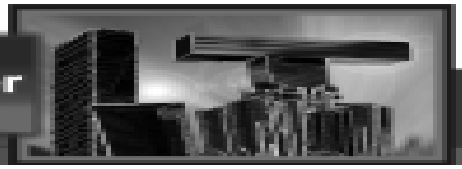
Application of Geographical Information Systems to Educational Plannig

Aronoff (1989) classified GIS analysis functions into four major categories. The four divisions are:

- Maintenance and analysis of the spatial data
- Maintenance and analysis of the attribute data
- Integrated analysis of spatial and attribute data
- Output formatting

In view of this classification, the GIS technology can be used for the following educational planning operators:

- Finding the coincidence of factors in geographical locations before deciding appropriate locations for educational institutions. At times it is essential to locate educational institutions in towns, with the following geographic features, plain or lowland, thickly populated, good road and rail transport network, good water supply, stable electricity supply and adequate suitable sites for locating schools.



- Updating geographic information on existing school lands in a district, so as to determine extent of infringement.
- Determining catchment area for schools, that is, geographical regions from where a school is to draw its population.
- Location-allocation analysis of educational resources, as such resources have to be shared in the most equitable and fair way.
- Determining the most efficient routes for itinerant teachers and school supervisors.
- Illustrating level of educational development in geographical regions for the purpose of comparison.
- Providing on-line information about each school land parcel to facilitate the processing of projects.
- Monitoring workloads, development activity, and financial information for planning, budgeting and fiscal control.
- Automating school map efforts done manually.
- Provision of automated support services, such as public notices and advertisement.

Other Capabilities of Modern Information Communication Technology

The Geographical Information System is an Information Communication Technology. The world has witnessed a lot of development in information communication technology. The innovation of computers and internet connectivity is a major development. According to Schroeder and Behaler (1996) ,the modern information communication technology enables us to be able to:

(i) transmit both voice and data simultaneously,

(ii) perform conversion of data among different or incompatible systems, allowing equipment from different vendors to communicate,

(iii) control local area networks from within a private brand exchange switchboard,

(iv) provide voice and electronic mail,

(v) perform asynchronous and synchronous transmission simultaneously,

(vi) find the least costly route for communication,

(vii) switch digital transmission without the use of modems,

(viii) enjoy data security by requesting and maintaining security access codes,

(ix) connect digital signals to high speech circuits,

(x) select different host computers or destinations without the need to re-arrange cables, and

(xi) have access to teleconferencing facility.

Conclusion and Recommendations

Geographical information systems (GIS) is a suitable tool for educational planning as there is a geographic dimension to education. Educational services are provided over space. Educational data are spatially referenced. Hence, the conventional management information systems may not be useful for processing of certain educational data, because of its geographic nature. Premised on the geographic nature of educational data, this paper recommends the adoption of GIS technology for the planning of education in Nigeria. Policy makers should also take cognizance of this fact and initiate policies that would favour compulsory adoption of GIS technology for planning in the education sector. GIS technology should be integrated into teacher education curriculum, as its knowledge is very essential for teachers who are often involved in micro planning.

REFERENCES

Aronoff, S. (1989) *Geographic Information Systems: A Management Perspective.* Ottawa, Canada WDL Publications.

Eastman, J.R. (1996) *Idrisi for Windows: User's Guide, Version 1.0* Worcester, M.A., USA: Clark University.

Fabunmi, M. and Fabunmi, B.A. (1998) Alternative Information Systems for School Mapping" *African Journal of Educational Management*, Vol. 1, Nos. 1 & 2.

Fabunmi ,M.(2004) *Perspectives in educational planning. Ibadan; Odunprints.*

Federal Republic of Nigeria (1998) *National policy on Education.* Yaba-Lagos: NERDC

Federal Republic of Nigeria (1999) *Constitution of the Federal Republic of Nigeria.* Abuja: Federal Government of Nigeria.

Romos, E, Schroeder, *A. and Behelar, A.(1996). Computer Networking Concepts.* London: Prentice Hall.

METADATA: The Key to Data Management in GIS Technology

Olatunji Babatunde 'Iekan

ABSTRACT

The growing availability of data from many different sources has helped GIS technology become more useful and widely adopted. With this development, the task of managing GIS data has become increasingly difficult. However, metadata makes spatial information more useful by making it easier to document and locate data sets. With metadata support, data producers can publish information about data and data consumers can search for the data they need. This paper discusses the metadata concepts and its importance in managing GIS data. More so, it focuses on the available metadata standards (as defined by standard organizations) to guide metadata creation process and provide useful information on set of tools that can assist practitioners in proper implementation process.

1.0 INTRODUCTION

The world is in the midst of an information explosion, but with increasing amount of data being created and stored there is a great need to document data for future use. Studies have established that although both government and society recognize the value of geospatial data, the effective use of geospatial data is inhibited by poor knowledge of the existence of data, poor information documentation of data sets, and data inconsistencies. Once created, geospatial data can be used by multiple software systems for different purposes. Given the dynamic nature of geospatial data in a networked environment, metadata is therefore an essential requirement for locating and evaluating available data. Metadata can assist the concerned citizen, the city planner, a student, or the forest manager find and use geospatial data, but they also benefit the primary creator of the data by maintaining the value of the data and assuring their continuous use over time. Some twenty five years ago, humans landed on the moon. Data from that era are still being used today, and it is reasonable to assume that today's geospatial data could still be used in the year 2020 and beyond to study climate change, ecosystems, and other natural processes. Metadata standards will increase the value of such data by facilitating data sharing through time and space.

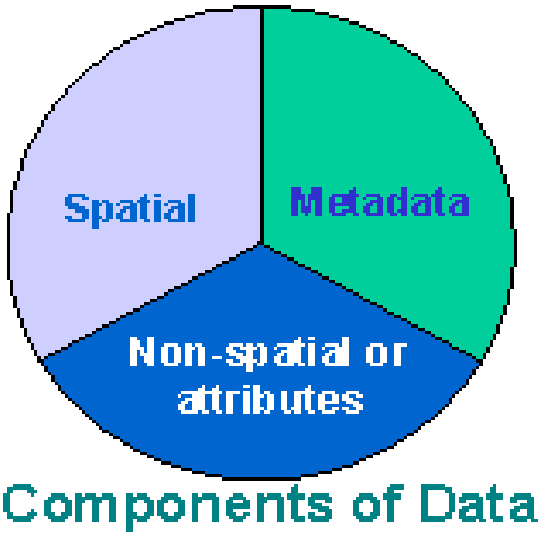
Why Creating Metadata?

There are a number of significant benefits in creating and maintaining metadata. These include:

- Metadata helps organize and maintain an organisation's investment in data and provides information about an organisation's data holdings in catalogue form.
- Coordinated metadata development avoids duplication of effort on existing data sets.
- Users can locate all available data set relevant to an area of interest.
- Collection of metadata builds upon and enhances the data management procedures of the geospatial community.
- It helps data providers in advertising and promoting the availability of their data and potentially link to on line services that relates to their specific data sets.

2.0 What is Metadata?

A traditional description of GIS data is that it is more than a computerized map, and it is more than just a database: it is both information about spatial location and relationships (coordinates, topology) and attribute (descriptive) information about geographic features in database format. However, this description is incomplete without metadata. GIS data can be described as having three components: spatial information, attribute information and metadata information [7].



Metadata is one of the three components of geospatial data

2.1 Definition

Simply defined, metadata refers to "data about data." However, in the digital spatial data context, metadata can be referred to as:

1. A summary document providing the content, quality, condition, and other appropriate characteristics of a data set.
2. "Metadata is simply documentation for a digital geospatial dataset. It is a text document that describes the who, what, when, where, why and how questions about the data, so that a potential data user can decide whether or not the data is appropriate for his/her use."

Metadata can be stored in any format such as a text file, Extensible markup language, or database record. Owing to its small size compared to the data it describes, metadata is more easily shareable. By creating metadata and sharing it with others, information about existing data becomes readily available to users. Thus, it makes data discovery easier and reduces data duplication.

For example, paper maps contain metadata, primarily as part of the map legend. In this form, metadata is readily apparent and easily transferred between map producers and map users. When map data are in a digital form, metadata is equally as important, but its development and maintenance often require a more conscious effort on the part of data producers and the chain of subsequent users who may modify the data to suit their particular needs

Metadata plays four major roles in spatial data management. These are: [1]

- Availability: data needed to determine the sets of data that exist for a geographic location.
- Fitness for use: data needed to determine if a set of data meets a specific need.
- Access: data needed to acquire an identified set of data.
- Transfer: data needed to process and use a set of data.

3.0 Metadata Standards

Metadata standards are guiding principles developed through a consultative process to provide a basis from which to develop national or discipline oriented profiles. These standards are aimed at providing a common set of terminology and definitions for the documentation of digital geospatial data.

In recent years, standards have been defined by official standard organizations to ensure a degree of consistency in metadata creation and management. Three major metadata standards exist or are in development that are of broad international scope and usage and provide detail for all levels of metadata mentioned earlier:

1. **The Content Standard for Digital Geospatial Metadata (CSDGM)**
2. **The International Standard Organisation standards(ISO)**
3. **The Comit Europe de Normalisation (CEN) Pre-standard**

However, for the purpose of this research work, consideration will be narrowed down to the CSDGM.

3.1 The Content Standard for Digital Geospatial Metadata (CSDGM)

This standard was developed in the United States of America and approved by the US Federal Geographic Data committee in 1994. It has been adopted and implemented in the United States, Canada, and the United Kingdom. It is also in use in South Africa, Latin American Countries and Asia.

The Content Standard for Digital Geospatial Metadata (CSDGM) <http://www.fgdc.gov/metadata/contstan.html> is organized into seven major sections.

Each of these seven sections contain a hierarchy of data elements and compound elements that define the information content for metadata to document a set of digital geospatial data.[1]

Metadata =

1. **Identification Information**
2. **Data Quality Information**
3. **Spatial Data Organization Information**
4. **Spatial Reference Information**
5. **Entity and Attribute Information**
6. **Distribution Information**
7. **Metadata Reference Information**

1. **Identification Information**: basic information about the data set.
   - Citation
   - Abstract
   - Purpose
   - Use Constraints
   - Person(s) to contact for questions about the data
2. **Data Quality Information**: a general assessment of the quality of the data set.
   - Source Information (Lineage)
   - Process Steps (Steps used to create the data set)
3. **Spatial Data Organization Information**: the mechanism used to represent spatial information in the data set.
   - Objects used to represent space in the data set E.g. Point, Vector, Raster.
4. **Spatial Reference Information**: the description of the reference frame for, and the means to encode, coordinates in the data set.
   - Data Projection
5. **Entity and Attribute Information**: details about the information content of the data set, including the entity types, their attributes, and the domains from which attribute values may be assigned.
   - Summary of the attributes
   - Definitions of the attributes
   - Values and units of measure for the attributes
6. **Distribution Information**: information about the distributor of and options for obtaining the data set.
   - What format the data are in (Shapefile, Image, etc.)
   - How the data are compressed (.tgz, .zip, etc.)
   - URL to download the data
7. **Metadata Reference Information**: information on the current- ness of the metadata information, and the responsible party.
   - Person(s) to contact for questions about the metadata
   - When the metadata were created

3.2 Benefits of Metadata Standards

Standards provide a number of benefits, theses include:

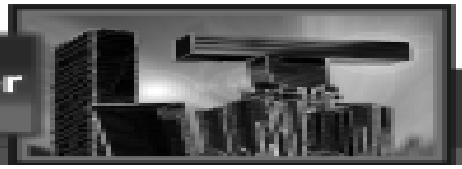
- **Standards provide a common set of terms:** with standards, there is no confusion about what is being communicated by a particular term. From one metadata record to the next, the terminology is the same.
- **Standards allow for quick location of a certain element:** if a standard is used, finding a specific piece of information in a metadata record will be much easier than if no standard is used.
- **Standards enable automated searches:** when standards are used, computers can be programmed to search and find useful data sets. This function of standards will become more important as more electronic data clearinghouses are built.

4.0 Metadata Tools

These are some of the tools for working with metadata. That is, the creation, validation, and publication of metadata. However, it must be noted that this tools are to assist and not totally replace human efforts. For example, no tool can check the accuracy of metadata or its proper conformance with standards. Consequently, some level of human review is required alongside the tool to have the desired output.

4.1 Metadata Creation Tools. [3, 5]

- **The ArcGIS ArcCatalog:** this is an application used for creating and authoring metadata and sending to a metadata service.



- **NOAA Coastal Services Center's ArcView® Metadata Collector Extension <http://www.csc.noaa.gov/metadata/download.html>:** this is a very useful tool for people who want to create metadata for their GIS data sets.
- Tkme: an editor for formal metadata <http://www.csc.noaa.gov/cgi-bin/goodbye.cgi?url=http://geology.usgs.gov/tools/metadata/tools/doc/tkme.html>. Developed by Peter Schweitzer of the U.S. Geological Survey (USGS), this editor was designed to simplify the process of creating metadata that conform to the Federal Geographic Data Committee's metadata standard.

Metadata Formatting and Reviewing Tools

- **Chew 'n Spit (CNS) <http://www.csc.noaa.gov/cgi-bin/goodbye.cgi?url=http://geology.usgs.gov/tools/metadata/tools/doc/cns.html>:** this is pre-parser used in checking written metadata to ensure it is properly formatted.
- **Metadata Parser (MP) <http://www.csc.noaa.gov/cgi-bin/goodbye.cgi?url=http://geology.usgs.gov/tools/metadata/tools/doc/mp.html>:** after running the CNS to make sure metadata is in proper order and format, this tool is used to check for technical errors. Once all errors are taken care of, this tool can be used to publish the metadata in a number of different formats (text, HTML, XML, or SGML).

5.0 Conclusion

Geographic data is perhaps not only the most important component of GIS, but are also very expensive to collect, store and manipulate. Hence the need for an effective data management mechanism called metadata. This paper has so far discussed the concept of metadata and presented available standards that can guide a sound metadata development process. It also highlighted some useful set of tools that can assist in this development process.

To this end, there is a great need for concerted effort towards the creation of our own indigenous metadata so as to assists in the successful application of GIS technology in Nigeria.

REFERENCES

1. Content Standard for Digital Geospatial metadata, <http://www.fgdc.gov>
2. David Hart and Hugh Phillips**, "**Metadata Primer — A "How To" Guide on Metadata Implementation", 1998, National states geographic information council.
3. "Metadata and GIS" , <http://www.esri.com>
4. Metadata Home Page, US federal Geographic Data Committee. <http://www.fgdc.gov/metadata/metadata.html>
5. <http://www.css.noaa.gov/metadata/metatools.html>.
6. <http://www.iso.org>
7. "What is metadata?", www.inside.uidaho.edu/tutorial/metadata/what is metadata.htm <http://www.inside.uidaho.edu/tutorial/metadata/what is metadata.htm>

Using Geographical Information System for Building Sustainable Nigerian Economy

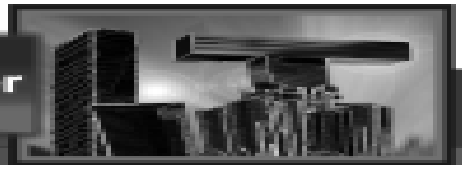
Iyekekpolor, A. E.

ABSTRACT

This paper sees Geographical Information System (GIS) as a computer based tool, which can be used as a resource technology to build a sustainable Nigerian economy that encourages employment. With it, individuals, organizations, government, schools and businesses, small and large-scale entrepreneurs, etc. can use it to solve their problems in the most innovative and creative manners. The paper highlights that the beneficial substances of GIS pose major challenges to the goings – on in planet earth today with a view to solving the world's pressing problems like: pollution, over population, natural disasters, deforestation etc, as they are also of critical geographic expression. Therefore, the main import of a Geographic information System, as said earlier, is a computer – based tool that is used for mapping and analyzing things and events which exist or happen on planet earth hence it aids decision – making through the interplay of database operations and statistical analysis for planning strategies, explaining events and forecasting outcomes. This is the reason why GIS performs tasks better and faster than the old fashioned manual methods. In this view, geographical information can serve as a reference tool which reveals longitude and latitude, rational grid coordinate, postal code address, street name, forest stand identifier, etc, for decision making processes as a way of solving a difficult problem in a business transaction. Better still, the potent influence of computers in shaping styles of thinking and thought, irrespective of subject areas, have made computer education to be introduced into tertiary institutions. This is the reason why it is opined that the common data for geographical information system are: base maps, environmental maps and data, business maps and data, and general reference maps. Hence, they work with the vector model and the raster model, which are very fundamental in the geographic information systems. As the system is integrated by geography, it stores information about the world from modeling global atmospheric circulation to trackling delivery vehicles and recording the detailed recordings of planning in all its ramifications. The paper, therefore, recommend among others that GIS should be taught in schools and that all stakeholders including the government should encourage the use of GIS as Nigeria is a large country that has a lot of human, material and mineral resources that are spread over the country.

INTRODUCTION

About two thousand years ago when the Roman Empire was at her peak, Roman roads heralded the growth of trade and commerce throughout Europe in manner that cannot be described with a flip of the pen. Stamper (2004: 14) illustrated that about one thousand years ago, the spice routes brought the cultures of the west and east together so much that spice trading increased in an unprecedented manner. Hence this brand of human activity has metamorphosed into Information and CommunicationTechnonolgy (ICT) as a result of the introduction of computers into the world. This is the reason why many enlightened people of the world are describing the earth as a "global village" hence; Nigeria cannot escape from it, but to cooperate despite her numerous limitations. Information Technology therefore is revolutionizing the way we live, think and work including our life and lifestyles. Akinde & Adagunodo (2001) opined that Information Technology has rendered "international boundaries irrelevant whereby several modern activities transcend international boundaries". It looks as if we now "live in a boundless world that is becoming a smaller place". Leon and Leon (2004) quoting Stewart (1997) observed that the emergence of the information age and the sudden ubiquity of information technology are among the biggest no, they are the biggest – stories of our time".



In the view, Geographical Information System (GIS) is a new technology in information technology, according to Leon and Leon (2004). They opined that GIS is a computer based tool which can be used as a resource technology and it poses a major chanllenge to the generation in the world today.

GEOGRAPHICAL INFORMATION SYSTEM FOR BULDING A SUSTAINABLE NIGERIAN ECONOMY

Nigeria, as a developing nation, is facing some major challenges today as in other parts of the world. Challenges, such as pollution, overpopulation, deforestation as a result of petroleum exploration , natural disasters, etc. are becoming new problems that demand peculiar solutions. Other new challenges like; finding the best soil fr growing cocoa, bananas, groundnut, etc, or siting new businesses or other human activities which have critical geographical dimension require geographic information system (GIS) to provide the necessary tool to enhance decision – making. This is why Rochester and Rochester (2003) added that map – making, business maps and data including general reference maps provided by GIS tool are better and faster that the old manual methods. Today, there is the need to need to teach GIS in Nigeria schools, colleges, polytechnics and universities in addition to other computer operations.

In this vein, GIS literacy and applications would enhance the employment of many Nigerians world-wide. The few industries, especially the petroleum prospecting ones, would have easy recruitment of Nigerians rather than going overseas to employ expatriate ones that would demand higher wages in foreign currencies. When this is achieved te Nigerian economy will be sustained. Hence Horbings (2001) opined that when a thing is sustainable, it means that thing would serve and go on for a long time to come. It would become and invaluable tool to the Nigerian economy especially now when many professionals in every field or career are increasingly aware of the advantages of thinking and working geographically. This is the reason why more people would like to work in urban centers or farms many kilometers from their homes.

Hanna and culpepper (2004) explained that a geographical information system (GIS) would empower someone to create maps, integrate information solve complicated problems, view scenarios, present powerful ideas and develop effective solutions. This is also in line with the views of Leon and Leon (2004) when they said individuals, organization, schools, governments and businesses that are seeing innovations in their pattern of production or service areas could use GIS as a creative way to solve their peculiar problems. Such problems, according to the authors, could be in the form of using data or maps in the following areas.

1. Base Maps: They include streets and highways; boundaries for census, postal and postal and political area, rivers and lakes; parks and landmarks, places name and USGS raster maps.

2. Business Maps and Data: They are data related to census or demography, consumer products, financial services, health care, real estate telecommunications, emergency preparations, crime, advertising, business establishments and transportation.

3. Environmental Maps ad Data: They are data related to the environment, weather, environment risk, satellite imagery, topography and natura resources.

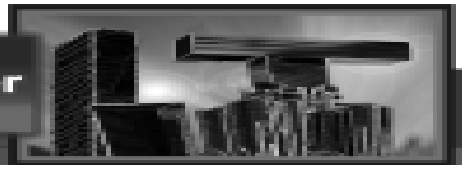
4. General Reference Maps: These are world and country maps and data that can be a foundation for individual's or organization database, Leon and Leon. (2004:32.4).

With the above information on data and maps, many authors, Leon and Leo (2004), Rochester and Rochester (2003) and Hohl (2004) opined that 'better information leads to better decisions" and this is true for geographic information system. They further stressed that a "GIS" is not an automated decision – making system but it is a tool that is used to query, analyze and map data in order to support a decision making process. This is the reason why it is the most common at the planning inquiries, aiding to solve territorial disputes as well as siting pylons in such a way as to minimizing visual intrusion."

COMPONENTS OF A GEOGRAPHIC INFORMATION SYSTEM (GIS)

Leon and Leon (2004:32:3) stressed that the workings of a GIS integrates fire key components like: hardware, software, data, people and methods. This is also in line with that of Chan and Lyon (2002) who added that there are other GIS related technologies like: desktop mapping, remote sensing and database management systems which aid decision-making processes when using a GIS.

a. **Hardware:** This is the computer on which a GIS operates. Leon and Leon (2004) opined that GIS software runs on a wide range of hardware types, they can be from a centralized computer servers to desktop computers used in either stand – alone or networked configurations.



b. Software: These are the tolls and functions needed to store, analyze, and display geographic information. The key software components are as follows:

(i) A Geographical User Interface (GUI) for easy access to tools.

(ii) Tools for the input and manipulation of geographic information.

(iii) A Database Management System (DBMS).

(iv) Tools that support geographic query, analysis and visualization, Leon and Leon (2004).

c. People: These are the professional personnel who manage the system and who develop plans for appling it to everyday nees.

d. Data: Data is very important in a geographical information system. Geographic data and the related tabular data can either be collected in-house or purchased from a commercial data provider.

e. Methods: A GIS is successfully operated through a well defined plan and business rules which are peculiar to the unique practices to each organization.

PROBLEMS

The following problems have been identified that go against the growth and development of a GIS in Nigeria.

1. **Corruption:** There is high rate of corruption in Nigeria. Nwogugu (2004: 135) opined that grand corruption and petty corruption have bedeviled the Nigerian economy like: political corruption, bureaucratic corruption, electoral corruption, embezzlement and bribery. This is why various structures are being attacked by the corruption virus hence valuables funds and resources initially earmarked for schools and other infrastructures are out rightly embezzled, misappropriated or other depleted severely.

2. **Lack of GIS Teachers:** There is an acute shortage of GIS teachers to handle the training of students' in tertiary institutions. These organizations that operate the GIS usually train their personnel either on job – training or are being recruited from overseas.

3. **Information Technology Market:** Developing nations like Nigeria and other African countries may remain information technology (IT) markets for the advanced countr, if the fail to evolve their own positive IT policies.

4. **Programmers in Developing Countries:** Programmers in developing nations like Nigeria and other nations may become idle people with time. This would happen if there is lack of interest and encouragement in the development of local software, and this might kill the IT industry n such developing countries.

RECOMMENDATIONS

From the foregoing therefore this paper recommends the following:

1. Federal, states, Local government as well as other stakeholders should encourage the computerization of their offices, businesses or parastatals with the inclusion of geographical information system (GIS) as Nigeria is a large country that has a lot of human, material and mineral resources that are spread over the geographical length of the country.

2. All stakeholders in the education industry should encourage the teaching and learning of GIS tools in te schools. This would help to create employment opportunities for beneficiaries or graduates after graduation.

3. All stakeholders empower the use of GIS in order to create better maps, integrate information present powerful ideas, etc, in organizations, government, schools and businesses, to solve their problems in the most creative and innovative arrays.

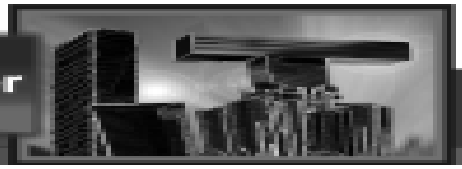
4. Nigerians should abhor all forms of corruption in order to evolve a better society.

CONCLUSION

This paper has looked into the geographical information system (GIS) as a tool for building sustainable Nigerian economy. It added that as the tool for decision-making is based on its geography, it would aid better mapping, solving business transactions, solving environmental problems and data including general reference maps, GIS would certainly encourage gainful employment.

REFERENCES

1. Akinde, A & Adagunodo, E. R. (2001). Information and Communication Technology in the



Administration of Colleges of education in Teacher Education in the Information Technology Age, (eds. Isyaku K., Anikweze, C. M., Maiyanga, A. A. & Olokun, M.) Garki – Abuja: National Commission of Colleges of Education (NCCE). Pp. 6 - 14

2. Chan, C Lyon, O. (2002). New Tools for Design Professionals. London, John Wiley & Sons.

3. Hinna, G. & Culpepper., D. (2004) GIS and Site Design: New Tools for Design Professionals, New York. Addison Wesley Inc.

4. Hohl, G. (2004). GIS Data Conversion: Strategies, Techniques and Management. Manchester: Onward Press.

5. Hornby, A. S. (2001).Oxford Advanced Learner's Dictionary of Current English. Oxford University Press Ltd.

6. Leon, A. & Leon, M. (2004). Fundamentals of Information Technology; New Delhi: Leon Vikas Publishing House, Pvt Ltd.

7. Rochester, C. T. & Rochester, D. R. (2003). Computers for People, Barton, McGraw – Hill.

8. Stamper, D. A. (2004) Business Data Communications, New York: Addison Wesley.

Application of GIS to Monitoring Changes in the Physical Environment Using Space Technologies - A Case Study of the Impact of Coastal Erosion on the Victoria Island, Lagos, Nigeria

Olatunde S. Ayodele

ABSTARCT

The biggest problem we have in studying changes in the physical environment is that of having an adequate scientific database. There is so much historical data available that must be considered. Presenting all of the data in a meaningful format is a big problem. With GIS, it is possible to manage changes in the environment. With a dynamic database management system that allows easy data update coupled with the capabilities of satellite remote sensing techniques to generate time series data and carry out change detection analysis of the landscape, GIS can be used to model and predict likely changes in the landscape. The most compelling reason for using GIS over standard paper maps is GIS power. The difference between a map and a GIS is the ability to query, ask complex questions of, the map data rather than to simply look at it. The problem associated with Victoria Beach erosion has been so enormous that the Federal Government had over the years commissioned various type of studies, both short and long term, to investigate the problem with a view to finding a long term solution to it. Some of these include investigations by Delft Hydraulic Laboratory, 1951, 1964 and 1975; Stanley Consultants 1968; Fawora Renardat Associates 1981; Webb 1960; Osoba 1971; Uboroh 1977; Ibe et al, 1984; Oyegoke et al 1984, 1985; Ibe et al 1986 (Onolaja, 1988). This study attempted to model the extent of coastal erosion on the Victoria Island by using evidence available from aerial photographs for two periods 1962 and 1976. This study is aimed at showing the usability of Geographic Information System using remote sensing as a mapping tool to the monitoring, modeling and control of erosion menace.

1.0 INTRODUCTION

1.1 BACKGROUND OF THE STUDY

Prior to 1908, erosion was not a worrying problem in the coastal areas of Lagos. The year 1908 is particularly significant when discussing the Victoria Beach erosion because this was when works on the Lagos harbour moles (break-waters) started. The works were completed in 1912. The moles were constructed at more or less right angle to the coastline to stabilise the (dredged) entrance into the Lagos habour. Although this objective was achieved, the moles triggered off a phenomenal beach erosion problem east of the inlet at Victoria Island. Since then, shoreline erosion is recorded to have been more than 1,500 near the east mole, decreasing to 800m about 1.5km away from the mole. (Onolaja, 1988).

The construction of the breakwaters produced the following effects viz:

(a) Altered the direction and mode of the littoral transport

(b) Created an imbalance in the rate of supply and loss of materials in the coastal region of the area which resulted in Victoria Beach eroding while the light house, west of the mole is accreting.

1.2 AIM AND OBJECTIVES OF THE STUDY

This study aim at modeling the extent of coastal erosion in Victoria Island based on evidence obtained from aerial photographs for two periods. (1962 and 1976).

The final products of this study include:

(a) Determination of the average rate of erosion for the area

(b) Determination of the area eroded.

(c) Predicting the coastline for various future dates.

1.3 JUSTIFICATION FOR THE STUDY

The 800km Atlantic Ocean Coastlines of Nigeria is a huge gateway to enormous resources of food, energy and mineral resources that are known to occur in the ocean beyond it. However, its role as a gateway is increasingly threatened by the scourge of marine

erosion which has guzzled up large expanse of coastal land and created a variety of serious environmental impacts including loss of life, property and income (A. C. Ibe, 1988). There have been a lot of attempts to study the character/nature of the coastal erosion especially the Lagos end of the Nigeria coast. A. C. Ibe, Awosika and others at the NIOMR has carried out considerable amount of work on the erosion menace of the Victoria Beach. Also students of the Department of Surveying at the University of Lagos have done some work on the same subject.

Although there has been a lot of work carried out on the studying of the erosion menace on the study area, to the best of my knowledge, the application of GIS is at its infancy.

"The biggest problem we have in studying coastal erosion is having an adequate scientific data base. There is so much historical data available that must be considered. Presenting all of the data in a meaningful format is a big problem" (Dickson, 1995).

With GIS it is possible to manage change in the environment. With a dynamic database management system that allows easy data update coupled with the capabilities of satellite remote sensing techniques to generate time series data and carry out change detection analysis of the landscape, GIS can be used to model and predict likely changes in the landscape. The most compelling reason for using GIS over the standard paper maps is GIS power. The difference between a map and a GIS is the ability to query, ask complex questions of, the map data rather than to simply look at it (ESRI ArcNews, Summer 1996).

1.4 THE STUDY AREA

Victoria Island is located approximately between latitudes $06^0 25'00"$ and $06^0 26'20"$ N and longitudes $03^0 24'00"$ and $03^0 28'00"$ E (Balogun 1995). It is located immediately east of the sea entrance into Lagos habour. On the south, it is bounded by the Atlantic Ocean, on the north by the Five Cowrie Creek, and on the east by the Kuramo Waters and Igbosere Creek. This is however, only an administrative boundary as the beach continues as a narrow strip between the sea and Lagos lagoon further inland merges at about 5 kilometres distance with the Ilado - Maroko beach (Ibe, 1988). See Figure 1

Victoria Island is part of the series of Barrier Lagoon Coast along the West African coast starting from Cote d'Ivoire and ending in Nigeria. In Victoria Island, heights of 1.5 - 4.0 meters above mean sea level are very common. See Figure 1.

The foreshore of Victoria Island is a low coastal plain with a few vegetated (prior to municipal development) beach ridges further inland representing earlier stands of the sea. The beach ridges are tightly spaced with an average relief of 1 metre. Old maps and aerial photographs indicate that the Island was a densely wood (mangrove) swampy approach to the active beach at the sea front before it was extensively sandfilled and developed as a high brow residential and commercial district.

Sediments on the island are medium to coarse sand (mean grain size 0.35mm) with an admixture of broken shells. The active beach is on average 50m wide and slopes rather steeply ($>5^0$) from the 1m (average) erosion scarp offshore-wards.

1.5 EROSION CONTROL MEASURES APPLIED FROM 1958 TO PRESENT AT VICTORIA ISLAND.

Many erosion control measures has been embarked on so as to solve the erosion and flooding problem threatening life and property on the Victoria Island. (See Table 1). Most of these, measures have not been successful while others are known to have created some other problems. On the 24th of August 1990, a devastating surge occurred and this led to about 7m of beach been eroded behind the Nigerian Institute for Oceanography and Marine Research (NIOMR) fence within 12 hours. On this day, waves were literally breaking at the edge of the Ahmadu Bello Way and almost cutting off the road.

The situation was so threatening that the Federal Government had to order an immediate stop-gap measure of beach nourishment. (Muritala, 1996). A consortium of dredging companies was immediately mobilised on the beach and the nourishment started in

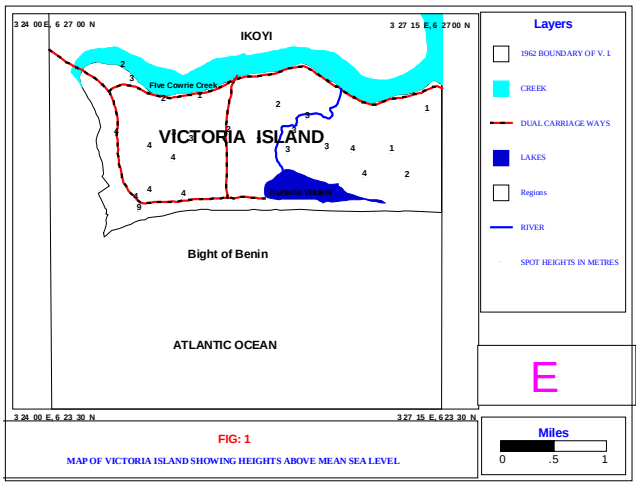


Figure 1

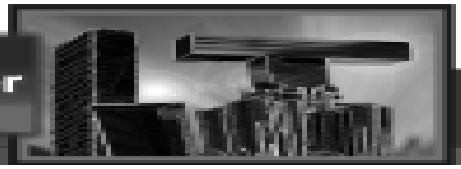


Table 1
Erosion control Measures applied from 1958 to 1998 on Victoria Beach, Lagos.

1958	Construction of a groin at the foot of the eastern breakwater, to avoid undermining of the breakwater
1958 - 1960	Dumping of sediment dredged from the Commodore Channel at the extremity of eastern breakwater, supplying an average of 0.66million cubic metres per annum sediment from the Commodore Channel to the beach. In between, in 1964, a 'zig-zag' timber groin (palisades) running parallel to the coastline was driven in some 26m from the shoreline.
1969 - 1974	Some artificial sand replenishment (but reliable records of quantities of frequencies are not available)
1974 - 1975	3 million cubic metres of sand dumped and spread on the beach.
1981	2 million cubic metres of sand dumped and spread on the beach
1985 - 1986	3 million cubic metres of sand dumped on the beach (before the work commenced, the culvert to the main boulevard parallel to the shoreline was already being undermined at some point by wave action).
1990 - 1991	3 million cubic metres of sand was dumped on the beach. Before the work started in August 1990 the entire sand dumped on the beach between 1985 - 1986 has been washed away in most places. The Lagos State Tourism fence along the main boulevard, as well as the main boulevard, Ahmadu Bellow Drive was already being undermined.
1994 - 1995	4 million cubic metres of sand was dumped on the beach. Before the work was completed in July 1995, the wave was virtually breaking at the edge of Ahmadu Bello Way and this render one lane of the road impassable.
1995 - 1996	Construction of groins behind NIOMR fence. This was completed in July 1996.
1997	2 million cubic metres of sand was dumped on the beach. *Source: NIOMR Victoria Island, Lagos.*

September 1990. Sand was dredged from offshore east of the Victoria Beach. A total of 5 million cubic metres of sand was pumped on the beach to raise the elevation and increase the width of the beach by 90 - 150m between the foot of the east mole and eastwards to the Kuramo Waters. This is supposed to act as a stop gap measure. The beach nourishment was completed about March 1991. (Balogun, 1995). In 1995, 1996 and 1997 other beach nourishment programmes have been implemented.

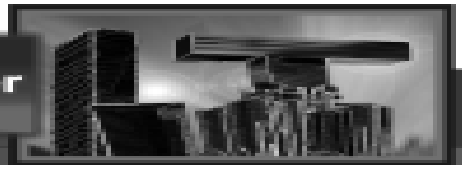
2.1 CONCEPTUAL FRAMEWORK

The suitability of satellite remote sensing as a mapping tool derives from its characteristics of comprehensiveness, timeless, repetitiveness, regularity, reduced dependence on weather and improved cost effectiveness (Ihemadu 1997).

Geographic Information System (GIS) is a decision support system involving the integration of spatially - referenced data for decision making in a problem solving environment. (Cowen, 1988).GIS works by combining database information systems with digital maps enabling us to analyse, query and explore data in a whole new way. Data and information are not the same thing. Data only becomes information when it can be used to answer a question or when it forms the basis of some action. GIS can integrate separate data sets, display them side by side, and give the decision maker the clear information needed to make the right decisions.

GIS makes it possible for data such as tide levels, flood protection works and population to be combined on a map background of a particular location. Maps can also be overlaid on one another, for instance the overlay of the maps of Victoria Bar Beach shoreline in 1960 and 1997 will clearly show the extent of shoreline recession in the time interval.

In this study, the aerial photographs of Victoria Island in 1962 and 1976 were overlaid. This gave the extent of the shoreline recession between the two period. Distance between the shorelines in the two periods was then taken at intervals and the mean of these distances was then calculated to get the average distance eroded between 1962 and 1976. This distance was then divided by 14 to get the average annual rate of coastline recession. The figure obtained was 6.5m which is within the figures of 8m and 4m given by Ibe on page 90 of his book "Coastline Erosion in Nigeria" as figures for the periods 1949 and 1957 and 1957 and 1962 respectively.



This figure of 6.5m was then used to predict for future dates 2000, 2010 and 2050 respectively using 1962 as the base year.

2.2 DEFINITIONS

2.2.1 Layer/Coverages

A layer (or coverage) is a map like collection that contains the geographic

definitions of a set of features and its associated attribute tables. Each layer will typically contain information on only a single feature type. One might have a soils layer, a road layer and the like.

With the GIS ability to manage differences of scale, projection and reference system, layers can be merged with ease, eliminating a problem that has traditionally hampered planning activities with paper maps (J. Eastman, 1995).

2.2.2 Overlay

Overlay is the process of combing two or more layers to create a new sets of information.

2.2.3 Buffer

A buffer zone is a region of specified distance around a map feature or features. Buffer zones are used to designate an area of influence or determine the proximity of one feature to another.

2.2.4 Splitting Map Features

The map/split command allows for the creation of new map features by overlaying one layer with another layer. With this command, the features in the input layer are split by the features in the overlay layer, and placed in a separate layer.

2.3 LITERATURE REVIEW

Recent years have witnessed an increasing number of publications supporting coastal erosion. Coastline erosion is not confined to Nigeria. From a global review of this problem, Davis (1986) estimated that three - fourths of the world's coastlines are retreating at a rate of 10cm or more per year while a third are retreating at a rate of more than 1m a year.

In Nigeria, coastline erosion causes serious concerns because it uproots coastal settlements, decimates agricultural and recreational grounds, destroys habour and navigational structures, dislodges oil producing and export handling facilities and upsets the hydrological regime in the coastal areas (Ibe, 1988).

Past efforts at abating the nuisance of erosion of the National Coastline consisted mainly of sand replenishment programmes using sand either from the foreshore or the back waters. These have failed to solve the problem as erosion has continued to devastate the coastline beyond each pre-nourishment limits. A reviewing of the situation by A. C. Ibe has traced the failure of this measure to an inadequate knowledge of the interrelationship between near shore ocean dynamics and shoreline evolution along the Nigerian coast.

Attempts to modify coastal changes to halt erosion require an appreciation of the factors at work in the coastal morphogenic systems, the pattern of change, the sources of the sediments, the paths of sediment flow and the quantities involved over a given period of time. Such knowledge enables the effects of various kinds of erosion control measures to be predicted and offset, or allowed for, in coastal engineering works (Bird, 1967)

Steve Dickson and Bill Duffy in their work on coastal erosion in Marine, U. S. A. compared historical photographs from the 1950's and 1960s to ones taken in 1991 to check erosion in the last several decades. Comparisons between common points on each set of photos showed positional accuracy of one metre or better. They converted all the coastline data to ARC/ INFO coverages. GIS tools allowed them to see both the obvious and subtle changes in coastline position over a 40 year time span.

Quadri, R. A. in his M. Sc. (GIS) project (1996/97 session, University of Ibadan) applied GIS to the modeling of the effects of global sea level rise on the Victoria Island.

This study is aimed at the evaluation and modeling of the combined effect of all factors causing coastal erosion on the Victoria Island.

3.0 METHODS OF DATA CONVERSION

3.1 PREPARATION OF THE BASE MAP

The aerial photographs of the study area S/S 279 CN70 - 11, scale 1:40,000 and S/S 279 & A 71W - 76716-3 scale 1:25,000 taken in 1962 and 1976 respectively were obtained from the Federal Surveys, Lagos.

The study area was then traced out using tracing film and drawing pen from each of the photographs. The one of 1962 which is on scale of 1:40,000 was enlarged by 160% to bring it to scale 1:25,000. The two transparencies were then overlaid and oriented to one another using common features. A composite map

was then produced showing the coastlines at the two period (1962 and 1976) amongst other features.

The resulting composite map was then geo-referenced by registering it using the Nigeria 1:25,000 topographical map sheet 279A N. E. 2 (Lagos N. E. 2) Edition 1 published by Federal Surveys, in 1985 as the base map.

3.2 DATA CAPTURE (MAP DIGITING)

The analogue paper map was converted into digital map by map digitizing. The map was digitized using two softwares - Atlas GIS and ArcView GIS.

4.0 DATA ANALYSIS AND PRODUCT GENERATION

4.1 DATA ANALYSIS

In this study, the map of the study area for two periods were available - 1962 and 1976 respectively. The study area boundary and coastline at the two periods were saved as VIEROS: 1962BD and VIEROS: 1962CSL; and VIEROS: 1976BD and VIEROS: 1976CSL respectively.

4.1.1 Area Eroded Between 1962 and 1976

To obtain the area eroded between 1962 and 1976, the two layers VIEROS: 1962BD and VIEROS 1976BD were overlaid and then splitted. The result was then copied to VIEROS: VER6276. Figure 4.2 shows the area eroded with boundary, lakes, river and dual road layers turned on.

4.1.2 Obtaining the Average Annual Rate of Coastline Recession

The 1962CSL and 1976CSL layers were turned on. The Atlas GIS distance tool was used to measure distances between the 1962 and 1976 coastlines at intervals. The mean of the distances obtained was then taken to give the average coastline recession between 1962 and 1976. The average annual rate was obtained by driving the figure obtained by 14 (the number of years 1962 and 1976). This gave a value of 6.5 metres. This value is found to be consistent with values obtained by A.C. Ibe when he compared photographs taken in 1949 and 1957 (8 metres) and 1957 and 1962 (4 metres). This figures was then used to predict for future dates.

4.2 PREDICTING THE COASTLINE FOR FUTURE DATES

The number of years between the future date and the study base year, 1962 is multiplied by the average annual rate of coastline recession of 6.5m to obtain the total distance the coastline would have receded by the future date.

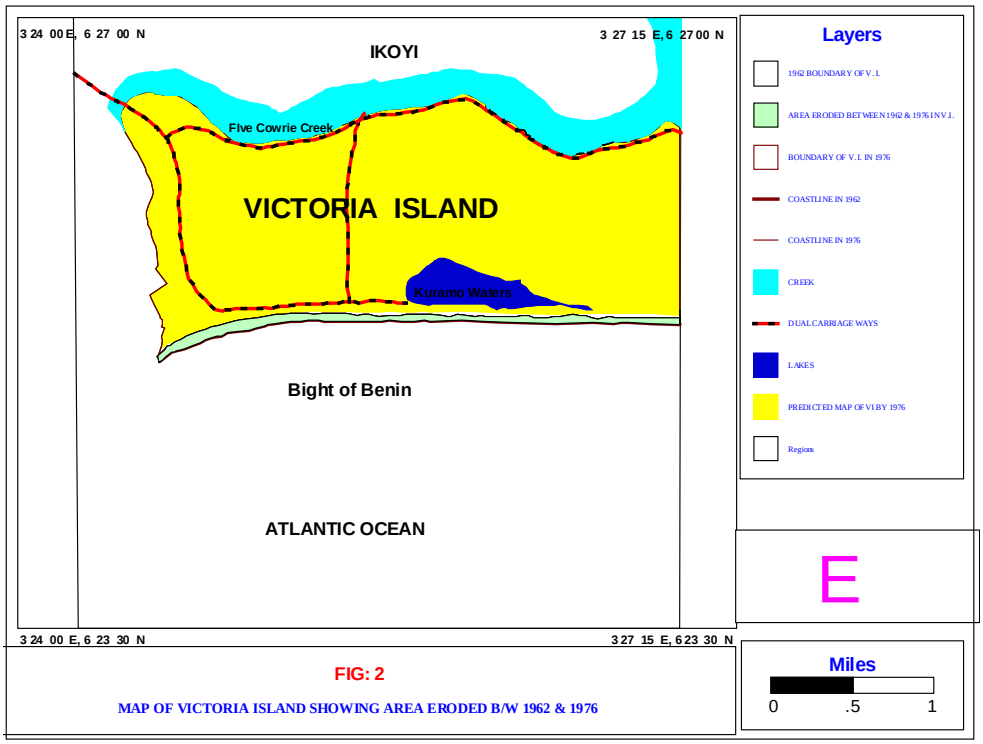


Figure 2

The resulting buffer zones created were then splitted from the base year map of 1962 to crate the map showing the Victoria Island by the predicted future dates.

To obtain the area of land that would have been eroded, the info button on the Button Bar is used on each layer and the area is copied. The area for each layer was then subtracted from the area for the base year to give the area of land eroded between 1962 the particular year. Figure 4.8 shows the map of VI at various dates.

4.2.1 Accuracy of the Predictions

To have an indication of the accuracy of our prediction, the boundary for 1976 was predicted. The output is stored in VI1976 layer. This is then overlayed with the boundary of VI in 1976 as obtained from aerial photograph and displayed as Fig. 6

4.2.2 Victoria Island by the Year 2000

The average annual rate of coastline recession of 6.5m was used to calculate the total distance that the coastline would have receded by the year 2000 using 1962 as the base year.

Receded distance = (2000 - 1962) x 6.5m

= 247m

This distance was then used to buffer around the 1962 coastline and the output saved as VI2000B.

VI2000B was then splitted from the map of VI in 1962 VI1962BD and the output put into VI2000 layer. Figure 3 shows this with some other layers for referencing purpose turned on. By the year 2000, approximately 127.9 hectares would have been eroded. This is approximately 12% of the total land mass in 1962.

4.2.3 Victoria Island by the years 2010 and 2050

The same procedure followed in 4.2.2. was used. The buffer zone was stored as VI2010B and the predicted map of VI2000 layer. By this year 161.38 hectares (15.5% of 1962 land mass) would have been lost to erosion. Map of VI by the year 2010 is shown in figure 4.

By the year 2050, 296.71 hectares (28.5% of 1962 land mass) would have been lost to erosion. The predicted map of V.I by the year 2050 is stored as VI2050 layer and is shown as Figure 5

5.0 DISCUSSIONS AND CONCLUSION

The research work was aimed at the use of technology to manage shoreline. It was aimed at been able to use GIS to predict the extent of coastal erosion and the coastlines over a given time period.

The biggest problem encountered was lack of an adequate scientific database. Most of the available

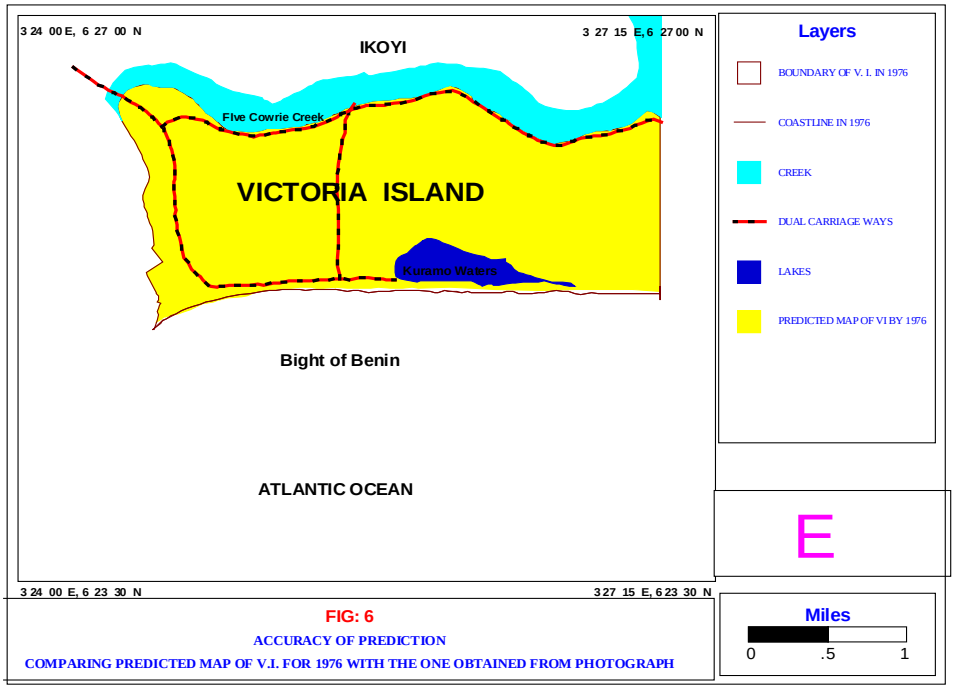


Figure 6

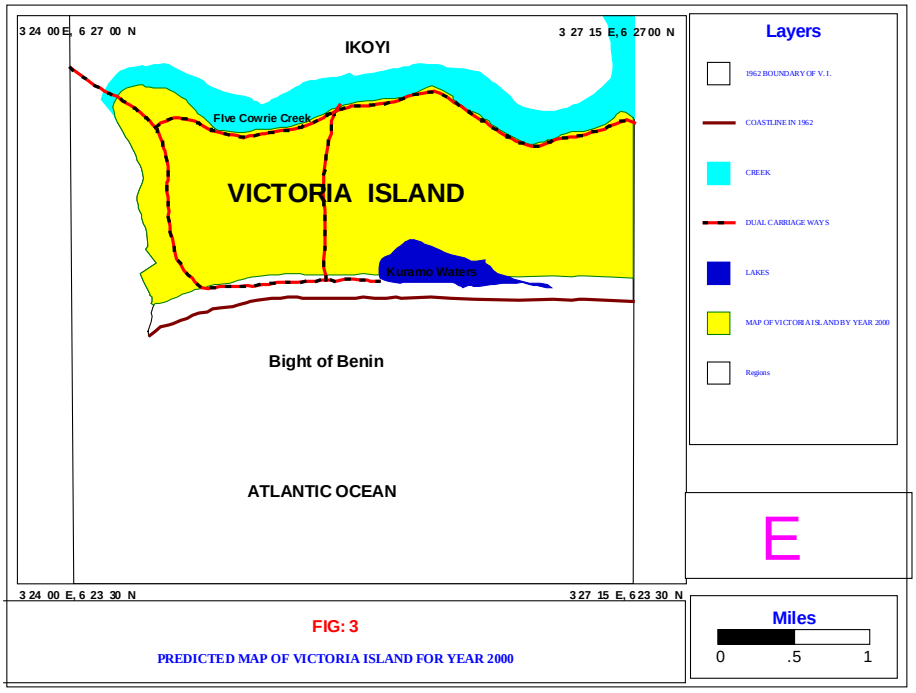


Figure 3

data were not in usable forms and formats. This was what led to our compilation of erosion data from aerial photographs of the study area taken in 1962 and 1976. This approach is pretty good, although we can not say that measuring one shoreline in 1962 and one in 1976 is all we need. To be absolutely certain of the behaviour of the shoreline, we must use several photographs taken over a long period.

The result of our findings revealed that about 12% of the 1962 land mass of the study area would have been eroded by the year 2000. By the year 2010, 15.5% and by the year 2050, 28.5% of the 1962 land mass would have been carted away by the sea. See figure 7.

In many parts of the world, coastal erosion is a severe environmental problem, reducing productive recreational land use, expensive prime lands etc. to nothing. To combat erosion effectively, extensive information on erosion status and erosion conditions is needed. This information can to a large extent be derived from satellite image data. Fortunately, there are available satellite imageries that have up to 0.6m resolution (digital globe) this will make it possible to have more precise models in future studies.

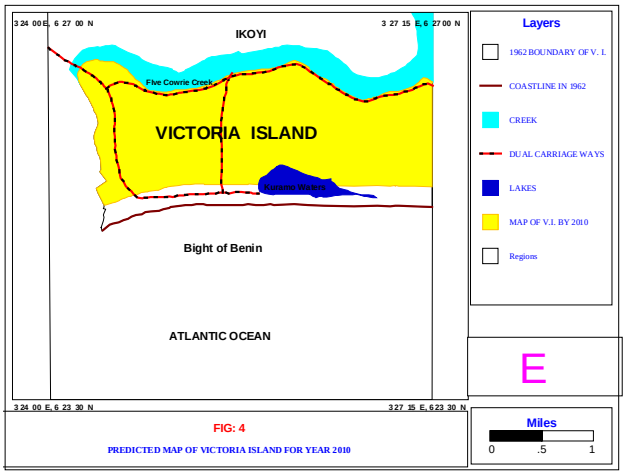


Figure 4

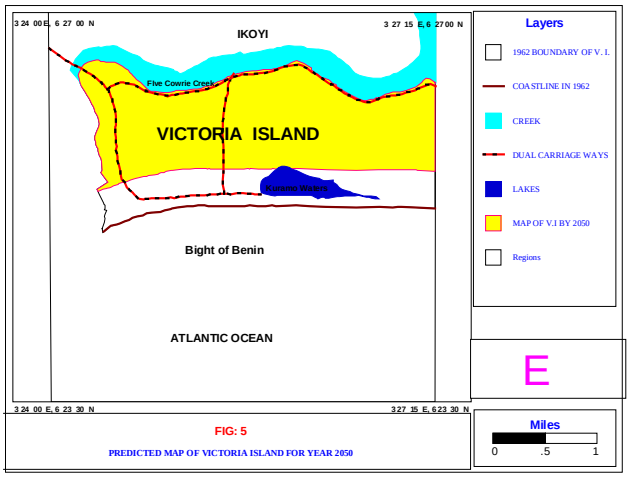


Figure 1

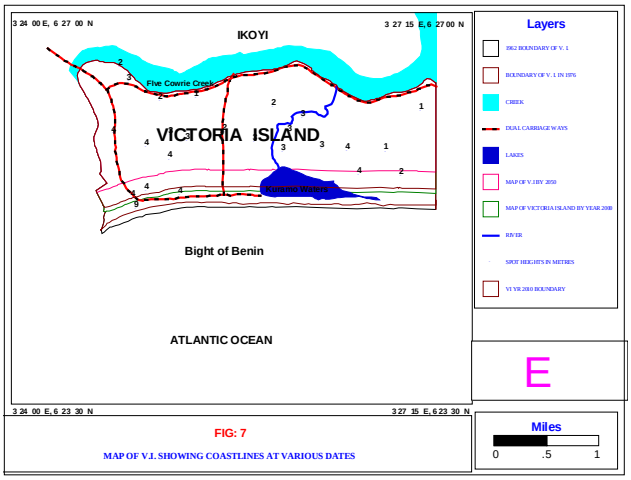


Figure 7

By visual interpretation of computer enhanced satellite imagery in combination with GIS, field and ancillary data, erosion maps on scales 1:25,000 – 1:250,000 can be produced showing:

- erosion conditions and indicators
- erosion changes over time
- erosion hazards

Together these forms a good basis for land development and conservation measures (Satellitbild, Sweden, 1988).

Today with the advent of better high resolution satellite imageries such as digitalglobe and ikonos (resolution of less than a metre), it will be possible to rapidly monitor rapid changes in the environment.

References

Adeoye, Anthony A. (1998) Geographic/Land Information Systems. Information Management Consultants, Lagos.

Awosika, L. F. - (1992) Coastal Erosion in West Africa: Causes, Effects and Response Options. *Paper presented at BORDOMER '92, International Convention on Rational Use of Coastal Zone, Bordeaux, France. 30 September - 3 October 1992.*

Awosika L. F., Ibe A. C., Shroader, Peter (Editors) (1993) Coastlines of West Africa. American Society of Civil Engineers 1993.

Ayodele O. S.. (1998)- Application of Geographic Information Systems To Modeling the Impact of Coastal Erosion (A Case Study of Victoria Island, Lagos) *M. Sc. (GIS) Thesis, University of Ibadan, Nigeria. (1997/98 Session)*

Burrough P. A. (1990) Principles of Geographic Information Systems for Land Resources Assessment. Oxford University Press 1990.

Dickson, Steve and Duffy, Bill (1995) - Using Technology to Manage a Shoreline *Published in Erosion Control, November/ December, 1995 and Reprinted in ESRI ARC News, Summer 1996*

Eastman J. Ronald (1995) Idrisi for Windows User's Guide Version 1.0. Idrisi Production, Clark University, U.S. A. (1991)

ESRI: "ArcNews" Summer 1996. *Published by Environmental Systems Research Institute, Inc. 1996*

Ibe, A. Chidi - (1988) Coastline Erosion in Nigeria. Ibadan University Press 1988

Ibe, A. C. (1986) Habour Development Related Erosion at Victoria Island. *Published in International Geomorphology Part 1, John Wiley & Sons Ltd. Pg 165 - 181*

Ibe, A. C. and Antia E. E. (1983) Preliminary Assessment of the Impact of erosion along the Nigerian Shoreline. NIOMR Tecnical paper No. 13 1983.

Ihemadu Surv (Dr.) - (1997) Impact Expectation of Current Satellite Surveying and Aerial Mapping Practices in Flood and Erosion Monitoring and Control. *Published in the Proceedings of the Technical Session of the 32nd Annual General Meeting and Conference of the Nigerian Institution of Surveyors, Uyo 7 - 9 May, 1997. (Pg. 58 - 65)*

Inegbedion L. E. (1990) Mathematical Modelling for Beach Stability Studies- A Case Study of Victoria island Beach. M. Sc. Research Project, Department of Surveying, University of Lagos. 1990

Muritala, Korede Lasisi - (1996) Monitoring of the Bar Beach for the Control of Coastal Erosion. B. Sc. Project, Department of Surveying, University of Lagos. 1995/96 Session.

Nwilo, P. C. (1997) Managing the Impacts of Storm Surges on Victoria Island, Lagos, Nigeria. In the Destructive Water: Water - Caused Natural Disasters, their Abatement and Control *(Proceedings of the Conference held at Anaheim, California, June 1996). International Association of hydrological Sciences Publication No. 239 (1997). Pg. 325 - 330*

Onolaja, L. (1986) A Review of Coastal Processes, Problems and Defences and the Federal Government Efforts in Combating the Problems of Flood and erosion along the Coastline. *(Proceedings of the National Seminar on Flooding and Erosion along the Nigerian and Similar Coastlines, 1986 pages 1-6)*

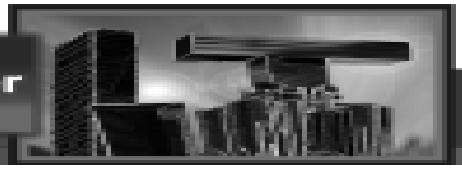
Onalaja, L. (1988) - The Effects of the Lagos Harbour Moles (Breakwaters) on the Erosion of Victoria Beach. *Published in the Design of Breakwaters". Thomas Telford Ltd., London 1988. (Pg. 129 - 155)*

Onolaja, L. (1989) Coastal Erosion Problems in Nigeria. *A Report of the Flood and ErosionControl Branch, Civil engineering Division, Federal Ministry of Works and Housing, Lagos. 1989 30pages.*

Orupabo, Sika (1989) Observations on the Erosion Problem at the Victoria Island Bar Beach *Seminar paper on the EEC funded Coastal Erosion Research Programme, University of Lagos. March 1989 28 pages.*

Peters K. O. & Raheem K. A. (1997) - Surveying for Coastal Zone Management. *Published in the Proceedigns of the Technical Session of the 32nd Annual General Meeting and Conference of the Nigerian Institution of Surveyors, Uyo, 7 - 9 May, 1997.*

Quadri, Raheem Adebayo (1997) - Application of Geographic Information Systems to Modeling the Impact of Sea Level Rise (A Case Study of Victoria Island, Lagos) M.Sc. (GIS) Project,



Department of Geography, University of Ibadan (1996 -97) Session

SSC Satellitbild - Products and Services - Soil Erosion Assessment (1998). *Published by Swedish Space Corporation.*

Star Jeffrey and Estes, John - (1990) Geographic Information systems – An Introduction Prentice - Hall International 1990.

A Framework for Knowledge-Based National Tourism Information System

Akinnuwesi B.A., Fajuyigbe O.

ABSTRACT

This paper takes a critical look at the operations of Nigerian Ministry of Culture and Tourism (NMCT) and considers it as, involving the processing of very large volume of culture and tourism related data and therefore focuses on data related to national tourism. National tourism information system constitutes a network of interrelated data and information processes. Data processing involves relating one data to another and when the number of data to be related to one another manually is very large; there could be an exponential growth, which users may not be able to manage manually. In view of this, this paper presents a report on the experimental study of an intelligent, interactive, user-friendly, knowledge-based system that carries out a stepwise analysis of tourism related data and information, filtering cognitive and emotional elements and apply both forward and backward chaining principles in making inferences concerning tourism related issues. This framework is believed to serve as a tool for collecting, storing and processing tourism related data and serves as a supporting artificial partner to human expert in the fields of tourism.

1. Introduction

Culture is a way of life in a society and tourism is one of the vehicles through which culture is appreciated. Tourism can be described as the business of providing holidays, tours, hotels, food services, entertainments and lots of others for tourists; therefore countries depend on tourism for income.

The benefits of tourism cannot be overemphasized; it develops the national economy by providing employment, enhancing foreign exchange earnings, provides infrastructure and increase international exposure and unique opportunities for building a strong national identity. It serves as avenue for appreciating the customs and traditions of various societies in the country. In view of these benefits, the Federal Government of Nigeria formulates tourism policy. The main trust of the policy is to make Nigeria a prominent tourist area in Africa, generate foreign exchange, encourage even development, promote tourism-based rural enterprises, generate employment, accelerate rural-urban integration and foster socio-cultural unity among the various regions of the country through the promotion of domestic and international tourism. The policy also encourages active private sector participation in tourism development.

In formulating the tourism policy by the Federal Government, there is the need for accurate information about tourism across the country. This information is to be provided by the various Tourism Boards across the country. The information is gotten from the various records kept by the personnel of these Tourism Boards. It is observed that records keeping and processing exercises carried out in many of the Tourism Boards today are done manually. This is cumbersome and not appropriated in the present age of Information Technology (IT). Thus, there is the need for a computer-based system to facilitate records storage, retrieval, processing and generating reports that can be used by management to make decisions and formulate useful policies to promote tourism in our society. In this paper, emphasis is on the design of a framework for a Knowledge-based System for managing the records of national tourism. This framework can be developed using a Database Management System (DBMS) to create and manage the knowledge base, which serves as the back-end of the system and a Visual Programming Language (VPL) to develop the front-end. It can be implemented in any Tourism Board across the country. In this paper the practical implementation is carried using Microsoft Access 2000 as the DBMS and Microsoft Visual BASIC as the VPL while the Ondo State Tourism Board serves as the case study.

2. Conceptualization of the Knowledge-Based System (KBS)

KBS is an Expert System, which is an area of Artificial Intelligence (AI). AI is the science of making machine to do things that would require intelligence if done by

human being [Elaine Rich, 1986]. Human activities such as writing books, computer programs, common sense reasoning, natural language understanding or driving a car are said to be demanding a level of intelligence. Over the years, computer systems have been built to perform these tasks, thus such systems possess some degree of artificial intelligence.

KBS is a collection of computer-based programs, which store the factual and inferential knowledge of experts. They use structured and unstructured knowledge and are able to draw conclusions from given case data [Akinyokun and Uzoka, 2000]. The name of the proposed system is abbreviated as KNTIS (Knowledge-based National Tourism Information System). KNTIS is composed of the following components.

a. Knowledge Base

b. Inference Engine

c. Decision Support Engine

2.1 Knowledge Base

Knowledge is a key factor in the performance of intelligent system. Knowledge can be classified into structured (quantitative) and unstructured (qualitative) (Akinyokun and Anyian, 2001). The knowledge base of the proposed system is composed of both the structured and unstructured knowledge on tourism in Nigeria. The structured knowledge is concerned with the facts, rules and events of tourism, which are commonly agreed upon by the expert on tourism issues. The unstructured knowledge is the knowledge acquired through the experience had on tourism events. It is the knowledge acquired by good practice, guesses and judgments (Ermine, 1995).

In this framework, the knowledge base is conceptualized as a network of relations. A relation is a two-dimensional table containing tuples and attributes. The general form of relation is given by:

$$R[A_1, A_2, A_3 \ldots A_n].$$

The name of the relation is represented by R, the set $\{A_i\}$, i = 1,2,...n, represents the attributes of the relation R (Codd, 1970).

With the view of having a good knowledge of the operations at the Tourism Board, the following records were obtained at Ondo State Tourism Board and well studied to get all the facts needed to design the knowledge base for the proposed system.

a. Records of tourist attraction centers:

The board maintains information about the tourist centers in the state, which includes the location, descriptions, supervising officer, facilities, level of development, number of staffs involved and the average revenue generated from the center.

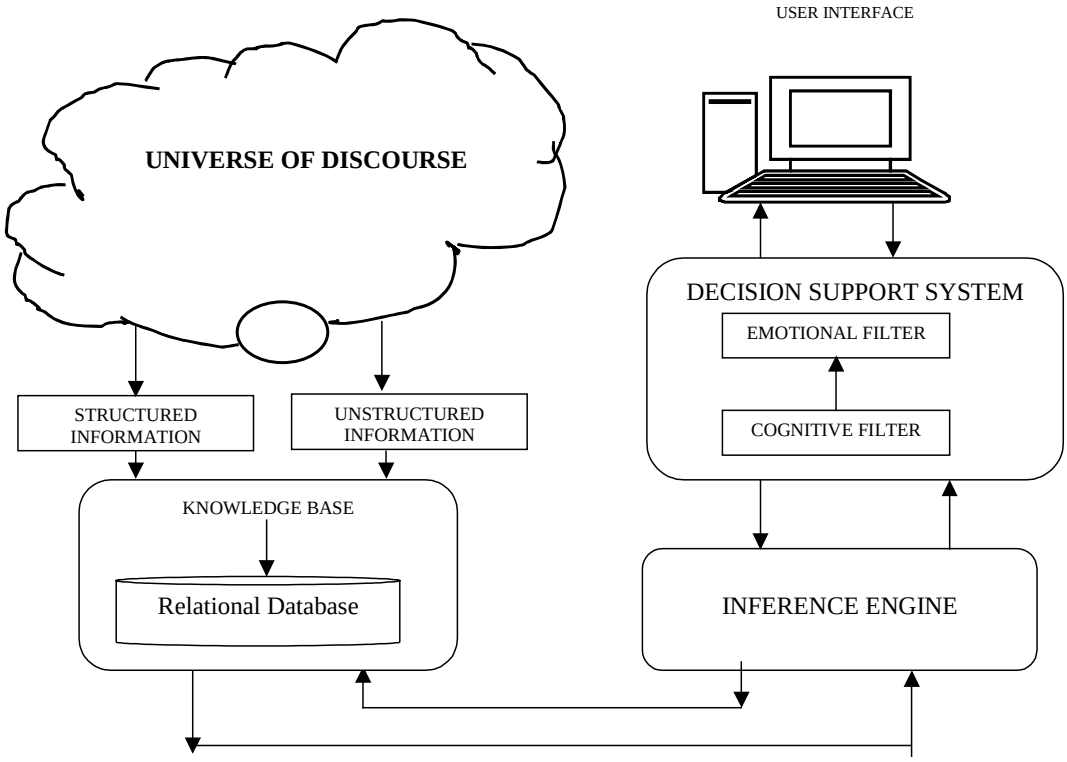
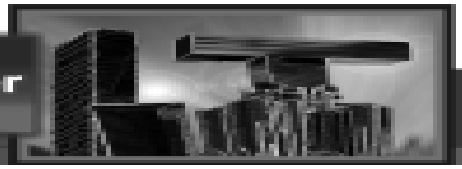


Figure 2.1: Conceptual Diagram of the Architecture of KNTIS



b. Hotels, restaurants and eateries:

Records are kept on the name of the hotel and their identification addresses, their grades which could either be International, national, rural, urban etc. The rates of booking, number of bed spaces and facilities available are also kept.

c. Records of registration fees for hotels:

Hotels, eateries and restaurants pay registration fees to the government at designated periods that is either monthly, quarterly or annually. A record is kept to this effect to ascertain this.

d. Demand notice forms given to hotels to pay registration fees:

Demand forms are forms that are issued to hotels, restaurants and eateries as notifications for payments to be paid and the expiration period of these payments.

e. Facilities available at the various tourist centers are documented.

f. Maintenance records:

The board keeps detailed documentation of maintenance processes in the tourist centers. It also specifies how contracts are awarded, the name of the contractor or company handling the job and the date the contract is awarded and the costs implication.

g. Personnel Records:

This contains information about the members of staff of the State Tourism Boards. Information kept includes name, age, contact address, date of appointment, schools attended, religion, marital status, salary grade level etc.

The abstraction of the above records is the set of relations listed below and these relations are considered for this study.

a. REGISTRATION [reg-no,name-of-institution,class-of-institution,amount-paid, date-of-registration, date-due-for-reregistration]

b. TOURIST CENTER[tourist-center–id-number, reg-no,tourist-center-name,town, local-government-area,center-description,level-of-development,name-of–supervising-officer, number-of-staff, revenue-generated].

c. RES-HOT-EAT[ResHotEat–id-number, reg-no, ResHotEat-name, tourist-center–id-number town, local-government-area, street-name, grade, rate-of-booking, number-of-suites, type-of-facility].

d. CONTRACTOR[contractor-id-no,contractor-name,contractor-addr,phone-no,tourist-centre-id-no,res-hot-eat-id-no]

e. MAINTENANCE[tourist-center-id-number, contractor-id-no, type-of-contract-awarded,date-awarded, cost-involved]

f. FACILITY[facility-id-number, facility-name, reshoteat-id-number, tourist-center-id-number, suite-id-number,type-of-facility,facility-quantity]

g. DEMAND_FORM[staff-id-number, staff-name, sex, religion, date-of-birth, marital-status, state-of-origin, contact-address, loc-govt-area, department, date-of-employment, post-held, grade-level, date-of-last-promotion, date-of-retirement]

h. PERSONNEL-BIODATA[staff-id-number,t-board-id-no,name,title,sex,home-twon,state-of-origin,nationality,address,place-of-deployment,dept,rank]

i. QUALIFICATION[staff-id-number, qualification, date-obtained]

j. RENOVATION[reg-no,date-last-renovated,date-of-present-renovation]

k. ESTABLISHMENT[reg-no,date-established]

l. TOURISM-BOARD[t-board-id-no,t-board-name,board-chairman,board-address]

m. VISITATION[reg-no,name-of-visitor,visitor-nationalitydate-of-visitaion,fee-paid,no-of-people,time-in,time-out]

n. REVENUE[reg-no,revenue-generated,date-considered]

The unstructured knowledge for KNTIS is obtained through interviews with people and administration of questionnaires. The inference engine carries out its evaluation exercises using the facts gotten.

2.2 Inference Engine

The inference engine provides the reasoning ability that enables KNTIS to search through the knowledge base by using the production rule mechanism and hence proffers solutions to various query transactions envisaged for the system. The inference technique can be either a forward or backward chaining type. In this study the forward chaining type is adopted, in which

conclusions are drawn based on available facts obtained through the combination of some decision variables on tourism issues. Some of the inference modules envisaged for KNTIS are stated below:

a. Evaluate the performance of tourist centres

b. Evaluate the standard of hotel, restaurants and eateries

c. Evaluate the revenue generated by tourist centres

d. Evaluate the operations carried out at Tourism Management Board

An example of the production rule employed to evaluate the performance of tourist centres is given below:

IF (total revenue generated in a month ≥ N500,000) AND (number of staff ≥ 50) AND (total visitation in a month ≥ 1000 people) THEN PERFORMANCE ? "EXCELLENT"

IF (N300,000 ≤ total revenue generated in a month < N500,000) AND (30 ≤ number of staff < 50) AND (700 ≤ total visitation in a month < 1000 people) THEN PERFORMANCE ? "GOOD"

IF (N100,000 ≤ total revenue generated in a month < N300,000) AND (20 ≤ number of staff < 30) AND (400 ≤ total visitation in a month < 700 people) THEN PERFORMANCE ? "FAIR" ELSE PERFORMANCE ? "POOR"

ENDIF

ENDIF

ENDIF

2.3 Decision Support System (DSS)

The DSS helps the decision maker to form preferences, make judgments and take decision. Decision making process has three components, namely; people, information technology (IT), and preference technology [Akinyokun and Anyian, 2001]. The people involved are personnel of Tourism Boards. The IT component consists of the computing system that handles the storage, processing, and analysis of relevant information. The preference technology consists of emotional and cognitive filters that help to clarify both the objective and subjective value judgments made when evaluating the possible consequence of alternative decisions. The cognitive and emotional filter carry out the inductive and deductive reasoning on the information content of the output reports of the inference engine.

Given below are some typical situations that would necessitate both cognitive filter and emotional filter:

DECISION 1: The decision to renovate a tourist center in preference of others that are equally qualified for renovation.

DECISION 2: The decision to establish hotels in a tourist center or community among others.

DECISION 3: The decision to rate tourist centers, hotels, restaurants, and eateries among others.

DECISION 4: The decision to promote the culture and ancient creatures of a community with the view of making it a tourist center.

DECISION 5: The decision to build tourist or recreational centers in a community.

DECISION 6: The decision to recruit the natives of a community to manage the operations of tourist centers, hotels and others institutions.

3. System Implementation

The framework for KNTIS was developed in Microsoft Access and Visual Basic environment. It was tested in a stand-alone computer with Microsoft Window 2000 as the operating system in a single user environment. The structure chart of the system is shown in Figure 3.1. The system is viewed in a top-down manner and access is gained to it by supplying a valid user name and password, both of which serve as access rights and control mechanism. If access rights are granted, the system presents a number of easy to understand menus and submenus. The menus and submenus contain valid transactions and inferences in the management of tourism data. The inference procedure is interactive and guides the user intelligently, but always leaves the final decision to the management.

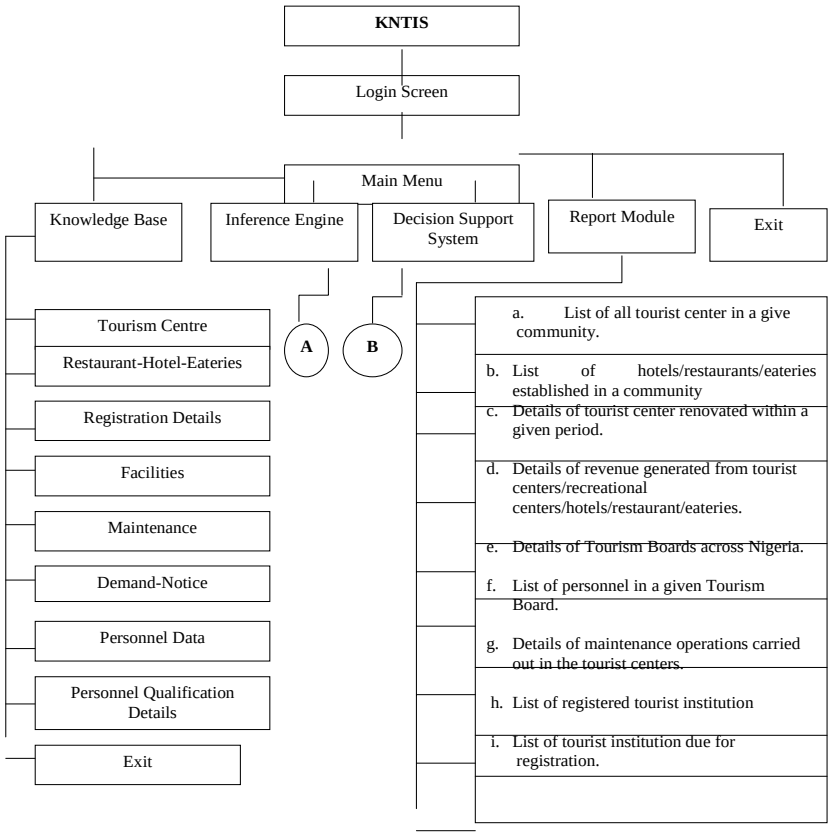
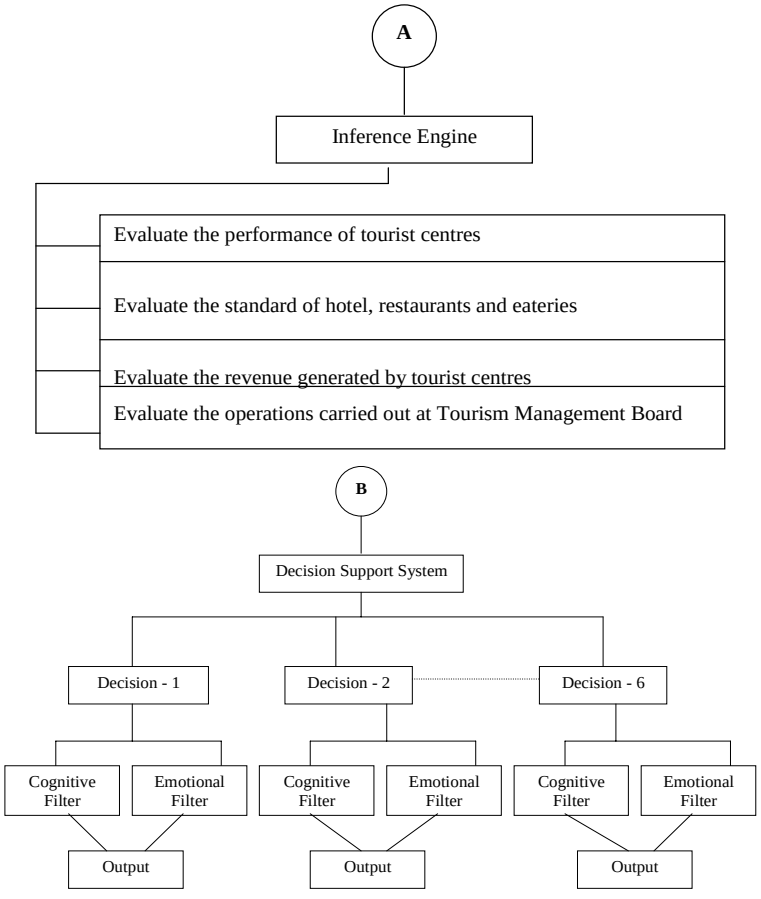


Figure 3.1: Structure Chart of the Proposed System



Dialogue and Menu Sessions of KNTIS

The first dialogue session takes the user through the login procedure. The user name and password are entered. Figure 3.2 represents the login window. Subject to verification and validation, authorization is granted and the system main menu is displayed. Figure 3.3 represents the window for the main menu. On clicking on "Exit", the user is taken out of the system, but on clicking on "Ok", the user is allowed access to the menu option and this leads to the second dialogue session.

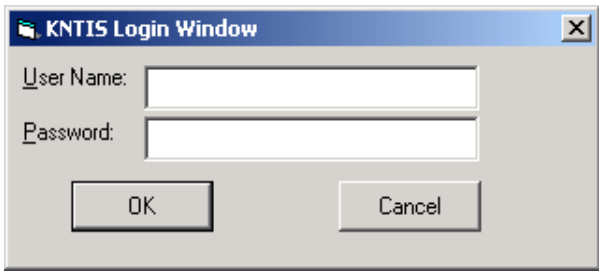


Figure 3.2: KNTIS Login Window

Figure 3.3: KNTIS Main Menu

If the "Knowledge Base" option is chosen from the main menu, a list of relations is displayed. Figure 3.4 represent the window displaying the list of relations envisaged for this system. On selecting the appropriate relation, the relevant form view is displayed. Figure 3.5 is the form view to input data into "Registration" file. The form view for each table gives the user opportunity to carry out the following basic update transactions:

a. Insert a new record

b. Locate an existing record

c. Delete an unwanted existing record

d. Edit an existing record

e. Undo an operation

f. Save the record in the database.

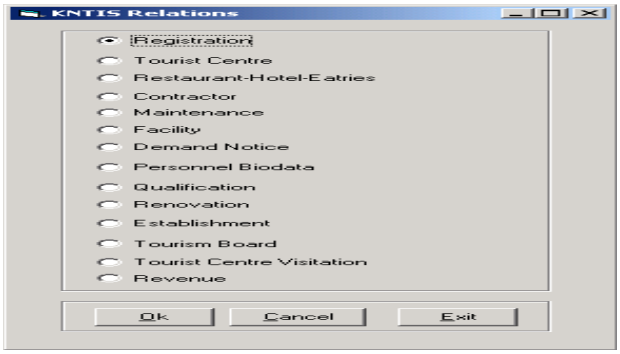
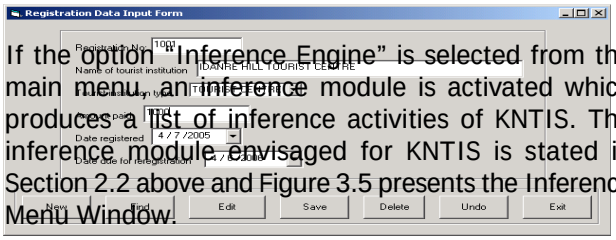


Figure 3.4: KNTIS Relations Window

Figure 3.5: Registration Data Input Form



If the option "Inference Engine" is selected from the main menu, an inference module is activated which produces a list of inference activities of KNTIS. The inference module envisaged for KNTIS is stated in Section 2.2 above and Figure 3.5 presents the Inference Menu Window.

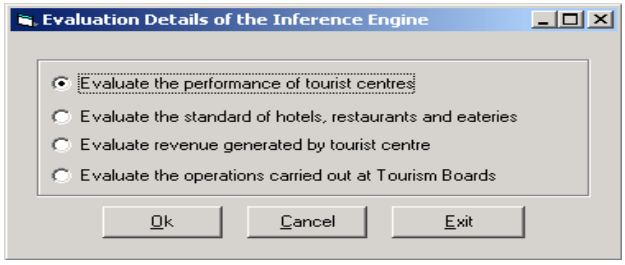


Figure 3.6: Inference Engine Window

If the option "Decision Support System (DSS)" is selected from the Main Menu, the window containing the list of decisions that can be taken by the system is displayed and this is shown in Figure 3.7. On selecting any of these decisions, a form will be displayed to allow the users to input both cognitive and emotional filtering data. This form is presented in Figure 3.8. Data input

will be processed by the system and the decision taken by the system will be displayed on the screen or printed. An example of the output of DSS is presented in Figure 3.9. This will be useful to the management while taken the final decision. For example, assume "Decision 1: the decision to renovate a particular tourist center in preference of others that are equally qualified for renovation". The user will have to respond to the following questions posed by the system:

a. What is the tourist center number?

b. When was it established?

c. How often do people visit the center (daily, weekly, monthly, yearly)

d. What is the average number of people that visit the tourist center in the stated period in (c) above?

e. How much revenue is generated from the center in the stated period in (c) above?

f. When was the last time that the center was renovated?

The responses to these questions serve as input variables for the cognitive filter of the DSS. Also the responses to the questions that will be stated below will serve as input parameters to the emotional filter of the DSS:

a. Any top government official form the community

b. Number government officials from the community

c. Management preference

d. Political group dominating the community

e. What ethnic group dominates the community?

The system processes the data supplied and make a decision, which can be adopted by the management. The management is left with the final decision. They may adopt the decision suggested by the system or ignore it.

Figure 3.7: Decision Support System Window

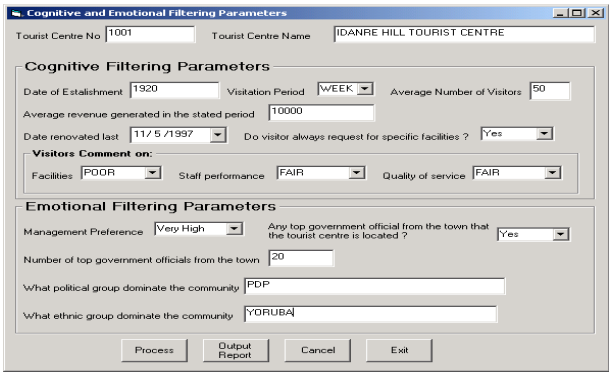


Figure 3.8: Cognitive and Emotional Filtering Parameters Input Form

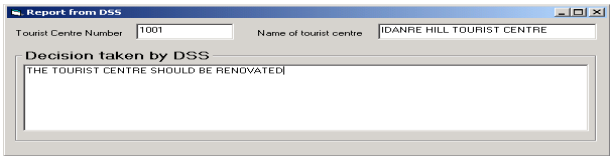


Figure 3.9: Report of DSS4.

Conclusion and Recommendation

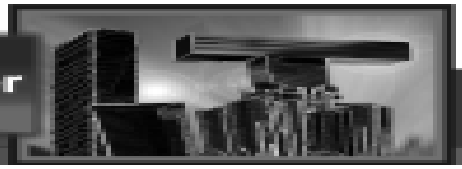
This work shows how a computer system can be a partner to the human expert. The computer-aided system facilitates fast and accurate information storage, retrieval and processing, which helps to reduce the problem of manual walkthrough of voluminous tourism data. The proposed framework enhances the management of data in the tourism boards. Also reports are produced quickly and this helps management to take decision at appropriate times.

This paper presents a framework, which can be worked on to meet the entire needs of the Tourism Boards and the Ministry of Culture and Tourism. The framework should be given further research in order to gain global acceptance and hence be useful to the Tourism Boards and Ministry of Culture and Tourism in Nigeria.

References

Akinyokun O.C. and Anyian T.N., 2001, "A Prototype of Knowledge-based System for Weather Monitoring and Forecasting", International Journal of the Computer, the Internet and Management, Vol. 9 No. 1, pp 13-23.

Akinyokun O.C and Uzoka F.M.E, 2000, "A Prototype of Information Technology Based Human Resource Syste", International Journal of the Computer, the Internet and Management, Thailand, Vol 8, pp 1 – 20.

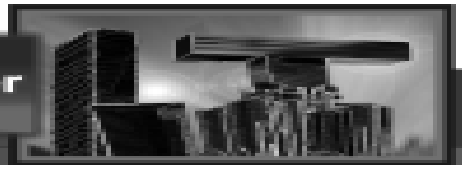


Codd E., 1970, “Relational Model of Data for Large Shared Data Banks”, Communication of ACM, Vol. 13 No. 6.

Elaine Rich, 1986, “Artificial Intelligence”, McGraw – Hill Book Co, Singapore.

Ermine J., 1995, “ Expert System: Theory and Practice”, Prentice Hall of India, New Dehli.

Uzoka F.M.E, 2003, “Knowledge Base System for Human Resource Evaluation in a University Environment”, PhD Thesis submitted to Department of Computer Science, Federal University of Technology, Akure, Ondo State, Nigeria, West Africa.



Open Source Hardware: A new perspective

Ojeyinka T. O., Uwadia C. O.

ABSTRACT

A lot of discussion has been going on concerning the policies, strategies and implementation of open source computing. Adequate emphasis had been given to open source software whereas little or no reference has been made concerning the hardware portion. Hence question that readily comes to mind is "Can hardware really be open?" This paper reviews the trends in open source computing starting from its introduction, history, policies and practices. It places emphasis on open source hardware and its implementation. The paper further discusses the advantages and disadvantages of this phenomenon and concludes by providing suggestions that are relevant to its success.

1.0 Introduction & Background

Definition

Recently, many application-specific integrated circuits (ASIC) involve deep submicron designs with many millions of gates in a single chip. Design times get prolonged as the number of gates continues to grow. The bad result of this trend is a long time- to-market and extreme cost. Core re-use seems to be the only technical solution to the problem. An advantage of this is the speed up in the design by enabling engineers to cooperate.

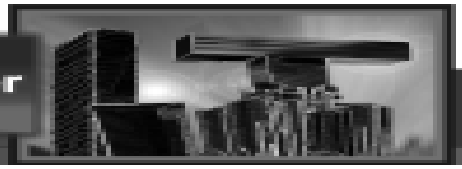
Available cores for integration are proprietary and must be purchased from established vendors which usually attract very high prices. These costs can be burdensome, especially for small design teams with limited funding. Proprietary cores are hard to integrate due to the multiplicity of incompatible design and test tools. Usually, proprietary cores lack adequate documentation, and the purchaser does not have access to the source, which makes the task of integration much more difficult, perhaps prohibitively so.

The definition of open source hardware is based on documentation and publishing of all necessary data about the hardware, the design specification, HDL files, simulation test benches, synthesis results, utilization instructions and interfaces to other systems, in which the its disclosure to the public is governed by the terms of GPL-like licenses [Parens 97]. Just as the source code in open source software is made available under the GPL, The EDA tools used to develop open hardware should also be open. This openness of resources is essential to allow the community reuse, develop and improve open designs.

Background

The first wave for the movement for open source hardware began in the 1960s when a group of students established the students for a democratic society forum for a participatory democracy based on community activism [Salem & Khatib 01]. It became obvious by 1970s that networked computers could be the basis for a new type of communication that could reinforce and extend communities but the existing computers could not be easily used for the job. The emergence of large scale integration Transistor-Transistor Logic (TTL) brought a large pool of individual hobbyist designers began to emerge. The design of 'Tom Swift Computer Terminal', Lee Felsenstein provided lead way for the first open source hardware in which he distributed the schematics as widely as he could. The initial home computer at this time was built on a culture of openness. IBM's commitment to open architecture in which the designs of IBM PC were published provided information and specifications which encourage many smaller companies and individuals to develop hundreds of hardware peripherals which comply to IBM standards. Without any pose their desired closed systems. The IBM's 5100 of 1978 was not open but the 1981 IBM 5150 (called the PC) made the architecture and executive code as public as possible, and encouraged individuals to write programs and create add-ons. This rule, created by Lee was tagged "You can play games, but you must help others to play as well".

While the first wave worked with sharing of designs based around commodity ICs, the second wave was entirely based on a set on people within the universities. By the late 1970s the size and complexity of chips grew rapidly and became enormous creating custom designs



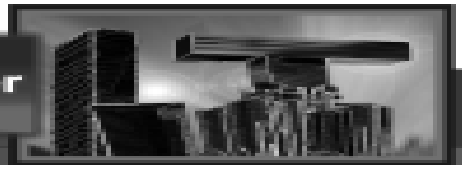
were demanded by chip designers while knowledge of all levels of the device, which was becoming increasingly unmanageable was needed. The two apparent requirements were (1) to separate the physical design of the chip from logical design and architecture, and (2) to start automating the process of deriving one from the other. Carver Mead and Lynne Conway [Clarke 01] invented a method which involves a technical part, involving the reduction of logical designs to simple diagrammatic representations with a direct, physically viable, semiconductor implementation. The APARNET's project which later became the Internet and the Free Software Foundation (FSF) and General Public License GPL of open source software on the internet helped in the sporadic spread of the method form the Xerox Parc to entire U.S. and then to Europe. Some numbers of design automation software were developed by universities like Berkeley and Stanford in US and Delft in Europe all of which were created under free licenses. Notable examples of such design automation software are Magic for layout, SIS and Espresso for logic minimization, Ocean for sea-of-gates designs [Ocean 97], Olympus for synthesis [Olympus].

By the early 90s, universities began to develop software under contract to the EDA companies, or sought to set up their own companies to commercialize their products. EDA software became closed source and either unavailable to the general public or tied to particular companies' products. From the mid 90s, after a period of nostalgia, technological and social changes once again drove a new flowering of free hardware design, which could also build on the experience of the past. The free hardware design had once again become prevalent for more than five years now under the same philosophy for which it was initially made.

2.0 Requirements for Open Hardware

The continued growth of Open hardware is based on a number of elements that drives its implementation. According to Bruce Perens [Parens 97] who defines a hardware design to be considered open source if its HDL source code was distributed with a license that (1) granted permission to freely distribute the source code and any hardware device based on it, and (2) granted the right to create derivative works based on the source code and distribute them under the same license. However the GNU General Public License (GPL) provides additional requirement that a special subset of the derivative works created *must* be available under the original license. Other requirements that would allow for successful implementation of open hardware are considered as follows:

- **Availability of open standards:** This concept is often put into practice in the software world by the Internet Engineering Task Force. The standardization process for a proposed internet protocol is customarily accompanied by an open source "reference implementation" **[Ristelhueber 99].** Hardware standardization bodies could benefit by embracing this practice, because it promotes both rapid and high quality implementations of standards. The benefits of providing a reference implementation are significant, but there are many other reasons why open source hardware design could be successful.
- **Availability of open hardware team or audience:** There must be availability of open hardware team or audience. This requirement is almost successful as the open source software. Open hardware community which was reinstated by Reinoud Lambert of the University of Delft provided the first free hardware design site to make a major impact was Open Design Circuits through which a large number of participants forming a community.
- **Open hardware development team:** Among the community of open hardware projects are the open hardware development team which provides the availability of yet another requirement for the existence of open hardware.
- **Availability of information on using the hardware** is considered to be another requirement. Although quite a number of developers sell hardware without providing information on how to use them, but this is actually a growing trend. If a company sells any device which interfaces to a computer without publishing the interface specifications, they then have a monopoly on interface software.
- **Availability of design for the hardware** had always been a problem for many devices since much digital hardware has never been open in such a way that allows the design to be published whereby other people can learn from them and improve on them. Nevertheless, exception exists with the SPARC architecture which gives companies with minority market share the knowledge of open sourced software as a tool be successfully imported to hardware.
- **Availability of design software for the hardware** forms another essential requirement. Processor manufacturers have always published their processor Instruction Set Architectures (ISA), and this had always made it possible to write free compilers. This trend, however, may change given an example of a giant processor vendor Philips



Trimedia Processor whose ISA is not available. It is a compressed VLIW whose compression algorithm is considered to be a trade secret. This same situation is true for most other types of hardware such that only the companies which own these secrets can create libraries or provide efficient design tools which use them. In the early period of some types of VLSI design and programmable logic devices (PLD), the internal structure and programming details were widely published in order to expand the market. The effect of this action made it impossible to create free software to make up for some of the deficiencies in these tools.

3.0 Implementation of Open Hardware: The Open Source Hardware process

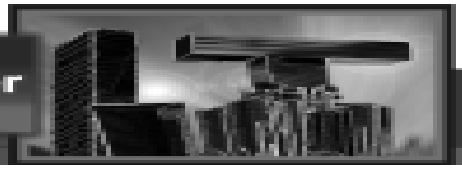
Hardware designs can be implemented application-specific integrated circuits (ASIC), custom silicon, field programmable gate arrays (FPGA), and complex programmable logic devices (CPLD). However, not all these programmable platforms fit into open-source hardware design. By using the hardware-software analogy the preferred programmable implementation platforms are the programmable logic devices such as PLDs, FPGAs, CPLDs and field programmable analogue array FPAAs. The analogy between software and hardware implementations applies to different aspects in the development process. Software programs run on general purpose processors, but open hardware designs fit on programmable logic devices.

From the processor's instruction set of a device, the equivalent assembly code is generated through software assemblers. The hardware synthesis tools generate a netlist of a particular device, using a digital or analog library. Binary code format is generated from an assembly program of a processor's instruction set using software compilers. The programming elements of an FPGA generate a bit-stream format from a netlist of device's component library. Optimization is done at this stage through dynamic reconfigurability [Seaman 01] of FPGAs in which performance of hardware designs is optimized using real-time dynamic loading and unloading of hardware components on the programmable logic array. This analogy between software and hardware execution and implementation phases helps prove the feasibility of adopting an open-source hardware strategy.

4.0 Tools for open hardware designs

- **Alliance:** Alliance is an open EDA system which provides a set of CAD tools and portable libraries for VLSI design developed at *Équipe Architecture des Systèmes Intégrés et Micro Électronique Laboratoire d'Informatique de Paris 6 Université Pierre et Marie Curie* [12]. It consists of more than 60 libraries all written in C language which contain the definition of several data structures and methods that allows users to describe circuits from the register transfer language (RTL) view up to the physical view. These data structures covered all the design flow and can be used by developers to implement CAD applications [Alliance]. Those libraries contain also parsers and drivers for several standards file format. Among the CAD tools for digital design flow featured by Alliance are VHDL Compilation and Simulation, Model checking and formal proof, RTL and Logic synthesis, Data-Path compilation, Macro-cells generation, Place and route, Layout edition, Netlist extraction, and verification, Design rules checking, etc ... Alliance uses the CAD top-down design methodology as described in [Ojeyinka et al 04] which includes Behavioral specification, RTL synthesis, and Place and Route.

- **Ocean:** OCEAN is a comprehensive chip design package which was developed at Delft University of Technology, the Netherlands. It includes a full set of powerful tools for the synthesis and verification of semi-custom sea-of-gates and gate-array chips. OCEAN is used to design the back-end of the design flow from circuit level, down to layout and a final working chip. Among the tools of OCEAN are SEADALI - an interactive layout editor and general interface for automatic and manual layout generation, MADONNA - an automatic placer, TROUT – an automatic router and connectivity verifier, FISH – a layout purifier for sea-of-gates and DRC, SPACE – a fast & accurate layout extractor, SLS – a logic level and switch-level simulator, SIMEYE - an interactive simulator interface which enables unequaled smooth work with SLS and SPICE, GHOTI – a circuit purifier for spice, CSLS XSLS, CEDIF XEDIF a write or read netlist formats. The design flow in OCEAN follows (1) functional design and (if possible) functional simulation, circuit generation, circuit simulation, layout generation (the generation of sea-of-gates layout description), layout extraction (the re-extraction of the transistor net list from the layout third quadrant), circuit simulation again (for verification of the layout and to check the effect of the parasitic in the layout). If necessary, the circuit description is modified and a new layout is generated. If the layout is satisfactory, the cell can be used as a son cell at the next level of hierarchy. The top level cell in the hierarchy contains the entire chip. The layout description can be converted into

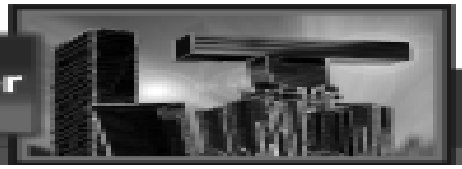


a file for mask generation and chip fabrication.

- **Magic:** Magic is a venerable VLSI layout tool, written in the 1980's at Berkeley by John Ousterhout, now famous primarily for writing the scripting interpreter language Tcl [Magic]. Due largely in part to its liberal Berkeley open-source license; magic has remained popular with universities and small companies. The open-source license has allowed VLSI engineers with a bent toward programming to implement clever ideas and help magic stay abreast of fabrication technology. However, it is the well thought-out core algorithms which lend to magic the greatest part of its popularity. Magic is widely cited as being the easiest tool to use for circuit layout, even for people who ultimately rely on commercial tools for their product design flow.
- **Debian GNU/Linux:** Debian GNU/Linux is a free operating system, developed by nearly a thousand volunteers from all over the world who collaborate via the Internet. Debian's dedication to Free Software, its non-profit nature, and its open development model make it unique among GNU/Linux distributions. Debian GNU/Linux is a free operating system, which now supports a total of eleven processor architectures with the addition of the IA-64 (ia64), HP PA-RISC (*hppa*), MIPS (*mips*, *mipsel*), and S/390 (s390) architectures, and this includes KDE and GNOME desktop environments, features cryptographic software, is compatible with the FHS v2.2 and supports software developed for the LSB. It now runs on computers ranging from palmtops to supercomputers, and nearly everything in between, including the latest generation of 64 bit machines. Debian GNU/Linux 3.0 features a more streamlined and polished installation, which is translated into numerous languages. The task system has been revamped and made more flexible. The *debconf* tool makes configuration of the system easier and more user friendly.

5.0 Business model for Open Source Hardware

- **Open source design implementations** allow companies to implement open source designs, sell them and pay royalties to original designers, according to their release license. A typical scenario of this type is exemplified by Elphel Inc. [Elphel] that offers Open source high resolution network Cameras. Among the range of cameras displayed by Elphel cameras ranging from high resolution and high frame rate to ultra high speed gated intensified cameras.
- **Competitors becoming co-developers:** is an evident condition as envisaged from the use of a GNU style open source license, requires the publication of derivative works, meaning that the value of the design, once open sourced, increases as the number of co-developers increases. Hence, no competitor may benefit significantly from the design without becoming a co-developer.
- **Third-party:** A design could be built from the ground up as open source by a design firm working under contract from one or more chip manufacturers. This development opens a number of business strategies: (1) Third party chip makers contribute additional funds in order to speed development or influence the specifications of the product. (2) Companies who have interest in the product under development can donate engineering resources to the project. (3) Furthermore, the design firm can leverage code reuse methodologies in order to reduce its engineering costs. (4) Lastly, although the source code to a completed design would be freely available, the design firm could generate significant revenue from support contracts as will soon be discussed.
- **Development and distribution of generic and customizable designs and cores** is a business model that enables companies under the GPL to pack sets of designs and sell the distribution just like Linux distributions an example of such a business is [20] in which, open source EDA tools and open source hardware designs distribution package are distributed by OpenTech for purchase. The package contains more than 260 open source EDA tools and 200 open source hardware designs, new programs and designs as well as some new changes and services. Another example is the sale of a CDROM of all the main open-source EDA software, and some open-source designs put together by Jamil Khatib of OpenIPCore, and is now ready for ordering at a cost price of $6.50 [Tiemann 99].
- **Technical support services** can be rendered by experts for open hardware designs intellectual property market. Using the same example given above, OpenTech provides a support program for some of their designs and tools to increase their use in the electronics community and to provide a complete solution for developers to use open source. This support model has also been very successful for Cygnus Solutions, which generates its revenue by providing commercial support for open source software development tools [Tiemann 99].



6.0 Open-source Hardware Organizations

This section introduces organizations that adopted the open source hardware model. Some organizations that are responsible for open hardware are discusses below:

- ***"Handasa Arabia":*** Founded by three young Arab engineers, the organization is a community of volunteer engineers worldwide that addresses the design of Open Source engineering systems' solutions. It states a mission of Arab nation development, whose essence is upheld to participate in the development of the "Hi-Tech" field globally while taking into account the Arab world needs. Its main objective is to design, reuse and integrate Open HW/SW IP cores under General Public License (GPL) compatible licenses, hence creating a deep commitment to the concept of openness represented by open-source hardware/ software IP cores. Handasa Arabia's main objectives are as follows: developing open-source engineering solutions as HW/SW IP cores, developing and integrating platforms using open-source engineering solutions as HW/SW cores, providing sufficient and well-written documentation and technical support, developing standards for developed open-source solutions, and spreading "Handasa Arabia" potentials in the "Hi-Tech" field by publications and exhibitions. Handasa's Projects include the OFOQ [Handasa c], the first Arabic PDA and NOUR [Handasa b], the bluetooth baseband IP core.

- **OpenCores:** OpenCores is an organization that is committed to the ideal of freely available, freely usable and re-usable open source hardware. The objectives of OpenCores are: to design and publish core designs under a license for hardware modelled on the Lesser General Public License (LGPL) for software, develop standards for open source cores and platforms, to create tools and methods for development of open source cores and platforms, to develop open source cores and platforms, and provide documentation for these cores and platforms. An open hardware design of a RISC architecture processor, open RISC1000 [OpenCollector], is used by some commercial companies that embed the design in their platforms. It is claimed to be a good alternative to FPGAs because it operates at higher clock rate and has a lower cost per unit price. It is also asserted to be a good alternative to standard cell implementations in terms of Non-recurring engineering (NRE) cost because designers can save up to $1million with a much shorter turn around time.

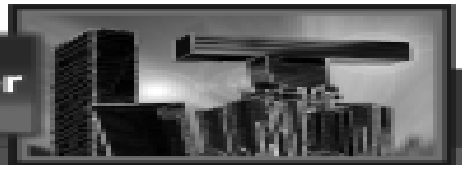
- **OpenSoCDesign:** Founded in 2004, by four highly qualified professionals from the R&D field from companies and universities the organization set out to increase the amount of open HW-SW and platform-based designs. Its services, based on our experience in the integration of HW-SW systems, are based on microprocessors, microcontrollers, platforms and FPGAs: OpenSoC System Design, OpenSoC IP Design and Verification, OpenSoC Open Source Hardware Integration which integrates systems based on Open Hardware Designs for users, OpenSoC GNU *toolchain* port which provides custom GNU *toolchain* for user's platform. The complete GNU *toolchain* for platforms includes *binutils*, *gcc*, *gdb*, and *eCos* operating system [OpenSOC].

- **gEDA:** The gEDA project was started because of the lack of free EDA tools for UNIX. The tools are being developed mainly on GNU/Linux machines, but considerable effort is being made to make sure that gEDA runs on other UNIX variants. The gEDA project is working on producing a full GPL suite of Electronic Design Automation tools. These tools are used for electrical circuit design, schematic capture, simulation, prototyping, and production. Currently, the gEDA project offers a mature suite of open-source applications for electronics design, including schematic capture, attribute management, bill of materials (BOM) generation, netlisting into over 20 netlist formats, analog and digital simulation, and printed circuit board (PCB) layout. Among the featured Free Hardware Project is PIC Binary Clock a hardware/software project that was designed using the gEDA suite of tools. A complete list of gEDA tools can be obtained in [Geda b] and tutorials in [Geda a].

Activity	Statistics	% of Total
Designers	860	72.76
Projects	210	17.77
Mailing Lists	18	1.52
News Posters	85	7.19
Sponsor Firms	9	0.76

Table 1: Sample statistics of some open-source organization [Salem & Khatib 01].

The following table provides statistics sample that could be considered as a performance indicator for open-source hardware involving Handasa Arabia, OpenCores and Opencollector activities.



7.0 Achievements of Open Source Hardware

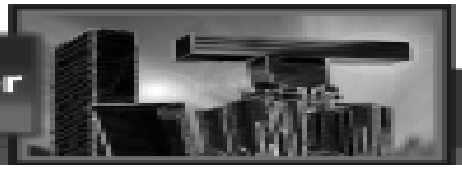
Most of the benefits of open source software can be ported to open source hardware since the both rely operate on equivalent terms of GPL. However, slight differences exist between the two in the sense that software is abstract and enjoys open source model more than hardware which is tangible. Having this in mind, nevertheless, there are still some areas of advantage of open hardware. Some added advantages of open source are discussed below:

- **Benefit of peer review** exists in open hardware designs where core design files and documentation are available for public scrutiny and professional modification. This makes the open hardware design model the best source for reusable cores, and facilitates an easy process of customization and adaptation for individual hardware needs. Users of these designs receive support from engineers and designers available within the open hardware community.
- **Promotion of uniform, high quality standards**: Open source is known to promote uniform, high quality standards. Companies compete for control which helps establish a standard design automation tool chain that could strongly influence EDA tool vendors to work towards compatibility.
- **Real circuit for learning:** One of the major advantages of open hardware is the availability of open circuit and system designs including the EDA tools and their implementations that are documented for public. This idea bring lots of prospects to research institutes and universities where students have the opportunities to learn and work with real circuits unlike in the past when students had to read various designs from textbooks without having the chance to use them due to the high costs of most commercial products.
- Furthermore, an open source strategy **eliminates many legal** issues that give rise to transaction costs, eliminates the need for cumbersome access control measures, and achieves the highest level of transparency possible in an intellectual property market.

8.0 Challenges faced by Open Source Hardware and Proposed Solutions

Although design of circuits and systems can be shared among developers via the Internet similar to sharing software files in open source software, there still exist some difficulty in sharing information about an implemented hardware device which cannot be passed across the internet due to its tangible nature. Software is abstract while hardware is tangible.

- **Dynamic hardware update:** Unlike software where code can be compiled and run during development, it is almost impossible to achieve dynamic hardware update in hardware reconfiguration of FPGA. This poses a challenge that needs to be resolved.
- **Design protection** for is necessary for original designer through licenses that protect his rights, according to particular terms and rules
- **The market or commercial success of open hardware** largely determines its success. Successful implementation of oh business models is intended to increase participation and confidence in open source hardware. Companies may oppose aspects of open source that generate alternatives for commercially protected products. The suggested solution is that companies might take advantage of open source as a way of bridging the gap for time and cost absorbed in R&D
- **Open hardware needs a lot of credibility and reliability** in order to instill confidence and participation among developers and co-developers. The solution to this is that designers produce high quality and completely documented designs just like Linux.
- **EDA tools are needed for a successful open hardware model.** Unlike the open software where availability of Linux operating system had provided a foundation upon which software development can be realized, open hardware needs powerful EDA tools most of which can only be found with big commercial EDA companies. A solution to this problem is the active development of EDA tools with modification feedbacks from other developers.
- **The high cost of fabrication and manufacturing** of ASICs posses a serious challenge to open source hardware since most of the hardware designs are based on programmable logic devices due to their cheap cost. However, solutions are on sight as OpenCore has shown its intention to start manufacturing its own chips that could be used for open hardware development.
- **The cost of verification and testing of design** before the final manufacturing is considered to be a common challenge to the success of open hardware. The problem may be alleviated as many big commercial companies join the open hardware race.



- **Problem of optimized open core for specific technology** makes it difficult to change design
- Other challenge that remains unsolved is the **precise licensing methods** should be adopted and used. There are indications, however, that the licensing issue will soon be resolved either by adopting the equivalent of open software GPL or a lesser GPL (LGPL).

9.0 The Future of Open Hardware

The OpenHW road map is divided into three main stages for new industrial and technological revolution by [18].

1. In the first stage several groups involves the design set of small boards that can act as generic boards that will serve different kind of systems and other applications. The designs will be simple and cheap and placed on the Internet so that anyone can download the entire design and documentation files, buy the components and build the circuit under available free "GPLed". These boards can be ranged from simple CPLD prototyping boards with simple interface, on board EPROMs, FLASHes or the CPLDs. while free EDA software developers will get feedbacks from designers which will help them improve their tools to match the requirements of the OpenHW designers
2. Before the end of the first stage, designers will publish OpenHW cores and OpenSource software for these boards. During this stage free EDA developers will collect feed backs from designers and improve their simulation, synthesis and verification tools.
3. In the third stage the hardware will be much close to the free software concept. In this stage new hardware design concepts will be introduced where anyone will have his own version of computer that he created from downloaded cores and software from the web.

The use of this idea will produce new free software-hardware compilers and new design methodologies.

10.0 Conclusion

From the discussion so far, open source hardware seems to be the solution to most of the problems associated with proprietary cores. Amongst its advantages lies the fact that each core will have a larger user base, which will ensure better support, better documentation and better implementation examples to work from. The source is available, so any developer can find out what he or she needs to know about the core. To some OpenHW organizations, there is no charge for using the core. Eventually, as cores and standards for them are developed, cores will become more standards-compliant than proprietary cores.

References

[Alliance] http://www-asim.lip6.fr/recherche/alliance/

[Clarke 01] **Clarke P.** Feb 2001 *Momentum builds for open-source processors* EE Times

[Copyleft] www.gnu.org/copyleft/gpl.html

[Elphel] http://www.elphel.com/

[Geda a] http://geda.seul.org/docs/current/tutorials/gsch2pcb/tutorial.html

[Geda b] http://geda.seul.org/download.html

[Handasa a] http://www.handasa-arabia.org/

[Handasa b] http://www.handasarabia.org/folder.php?id=11

[Handasa c] http://www.handasarabia.org/folder.php?id=17

[Khatib 02] Khatib J. 2002 *Design trend: A survey.* Geocities Mag http://www.geocities.com/SiliconValley/Pines/6639/docs/openip.html

[Magic] <http://bach.ece.jhu.edu/~tim/programs/magic/magic.html>

[Marshal et al 81] Marshall M., Waller Larry, and Wolff H. October 20, 1981 *Electronics* EEC http://ai.eecs.umich.edu/people/conway/Awards/Electronics/ElectAchiev.html

[Moller] Möller E. Dec 2004. *The Free Hardware Design movement: a smaller sibling to the Free Software movement* Boston, USA.

[Ocean] <http://cas.et.tudelft.nl/software/ocean/ocean.html>

[Ojeinka et al 04] Ojeyinka T. O., Oladipo O. F. 2004 *VLSI Design Methodology and Alliance CAD System.* Proceedings of the Nigeria Computer Society 8th International Conference, Abuja, Nigeria

[Olympus] <http://akebono.stanford.edu/users/cad/synthesis/olympus.html>

[OpenCollector] http://opencollector.org/

[OpenCore] *Opencores Road Map.* http://www.opencores.org/

[OpenSOC] http://www.opensocdesign.com/about.htm

[OpenTech] *OpenTech Cd-rom: About* http://www.opencores.org/projects.cgi/web/opentech/about

[Parens 97] Perens B., 1997 *The Open Source Definition,* OpenSource.org, U.S.A http://www.opensource.org/docs/definition_plain.html

[Pomerantz 00] Pomerantz G. M. 2000. *Business Models for Open Source Hardware Design*, Lyrisoft.com, U.S.A. http://lyrisoft.com/~gmp/papers/bmfosh-1.0.html

[Ristelhueber 99] Ristelhueber R. October 27, 1999 *Rapid chairman vows to remedy IP licensing lags*, EE Times.

[Salem & Khatib 01] Salem M. A. and Khatib J. I. 2001 *An introduction to open source hardware development*, EEdesign.com U.S.A. </exit?url=http://www.eedesign.com/>

[Seaman 01] Seaman G. 2001 *Free Hardware Design - Past, Present, Future*

[Solomon 03] Solomon J. B. May 2003. *The Art of Successful ASIC Design*

[Tiemann 99] Tiemann M., 1999 *Future of Cygnus Solutions*, *Open Sources*, O'Reilly and Associates, Inc, U.S.A

[Turley 02] Turley J. May 2002 *Open-Source Hardware Embedded Systems Programming,* Embedded.com U.S.A <http://www.embedded.com/>

Security and Open Source Software: A Critical Analysis

Olorunfemi, Temitope, Oladipo, Onaolapo Francisca

ABSTRACT

With the increasing interest and the rising popularity of open source software, the question of security arises. In particular, despite its prominent use in providing many aspects of the Internet's basic infrastructure, many still question the suitability of such software for the commerce-oriented Internet of the future. It is the objective of this paper to show that the proprietary/closed source software could be secure in theory but not in practice. We evaluated the suitability of open source software with respect to security which is one of the key attributes that tomorrow's Internet will require. It is also reported here that Open Source Software certainly does have the potential to be more secure than its closed source counterpart. Different arguments put forward, for and against open source security were analysed in relation to empirical evidence of system security from previous studies

Introduction

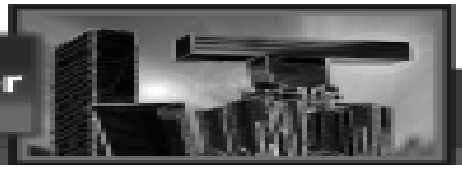
For many years now, open-source software has been used for carrying out critical tasks in Internet and Intranet. **BIND** is the dominating Domain Name Server, '**sendmail'** and '**qmail'** handle most e-mail, **'Apache'** runs over 50% of the world's web sites and the programming language **'Perl'** produces the active content. **Linux** and **FreeBSD** are used as the operating systems for even the most demanding sites, such as Yahoo, and often out-perform their commercial equivalents. It is worth noting that open source software can easily be integrated into enterprise-wide IT, including CORBA applications, directory services and security infrastructures [11]. What, then, about the security of open source software?

Open source software, by definition, is any program or application that is freely distributed, non-platform specific — and in which the programming code is open and visible [11]. There has been a lot of debate by security practitioners about the impact of open source approaches on security. One of the key issues is that open source exposes the source code to examination by everyone, both the attackers and defenders, and reasonable people disagree about the ultimate impact of this situation. To some, closed source means hidden, secret - and more secure. In reality, many of the most secure systems available today are based on the open source model [1].

This paper examines the concept of openness as related to Software security. We contend that obscurity may inconvenience blackhats a bit and help limit the number of potential attackers, but it works only so long as obscurity is maintained. Secrecy can be useful, but it is a fragile defense. We show that Closed-source products are not necessarily inferior; but, human nature being what it is, they often turn out inferior. Few people are as diligent as possible when they can easily conceal their shortcuts and mistakes.

What is Open Source?

Before we can intelligently discuss the differences between open source and proprietary software security, we need to clarify what the term really means. Many people equate "open source" with "free of charge," but that is not necessarily the case [2]. Open source code can be – and is – the basis for products such as RedHat and dozens of other commercial distributions of Linux that range in cost from a few dollars to a few thousand (RedHat Enterprise Linux premium edition lists at $2499 for Intel x86, up to $18,000 for IBM S/390). Open source" also does not mean "unlicensed." In fact, there are a whole slew of licenses under which open source software is distributed. Some of the most popular include GPL (the GNU Public License), BSD, and the Mozilla Public License [8]. The Open Source Initiative (OSI), a non-profit corporation, has developed a certification process for licenses. The name itself tells the story: open source software means the source code (the programming often written in C, C++ or even assembler language) is available to anyone who wants it, and can be examined, changed and used to write additional programs. This is in contrast to "closed" or



proprietary software such as Microsoft Windows, for which the source code is a closely guarded trade secret (except when it is leaked to the public) [8].

Open source software typically is viewed as software that meets the following criteria: free redistribution (licensees may resell or give away the software without paying royalties to the licensor); source code availability (the source code must accompany the software or be available on request); permission for derived works (licensees must be permitted to modify the licensed program); notice and attribution (licensors may require licensees to identify derived works as different from the licensor's original work); nondiscrimination (licensors may not discriminate against persons or fields of endeavor in licensing the software); license distribution (open source licenses apply to all users to whom the software is redistributed, without the need for executing additional licenses); and the absence of tying (licensors may not require that open source software be distributed with other software, nor may licensors place restrictions on other software distributed with open source software) [8]. One example of a widely recognized open source software system is the Linux operating system.

Security Through Obscurity?

Traditionally, secrecy has meant security. You lock up your house, your automobile, your valuables, and so on. In the software community, you "lock up" the programming source code as a means of securing it against hackers and competitors. Computer security is the process of balancing the security requirements of a computer system with requirements for price, usability, user training and reliability [1]. When considering open source software, it is natural to ask what impact the adoption of open source has on this balance. Vendors of proprietary software say keeping the source code closed makes their product more secure. This reasoning is based on logic; certainly you do not want to advertise what goodies you have in your house and where they are located to the neighborhood burglars. Open source advocates counter that this is merely a form of "security through obscurity," a concept that is generally dismissed as ineffective in the IT community. And certainly, by itself it would not protect you, as a homeowner or as a software vendor. Merely keeping quiet about your possessions might make it less likely that thieves will target you, but you would be foolish to leave your doors unlocked at night just because you have not distributed information about what you own. Keeping the source code closed might deter some hackers, but the large number of successful attacks against Windows and other proprietary software proves that it certainly does not provide any kind of high level of security.

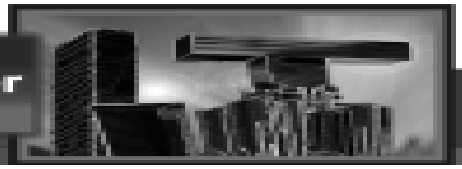
Closed Source Program can come Open

Early in 2004, it was reported that part of the source code for Windows NT 4.0 and Windows 2000 had been leaked to the Internet [7]. Files containing the code were posted to a number of P2P sites and were being eagerly downloaded. The available code comprised only a small portion of the entire code base for the operating systems, but the incident caused a great deal of consternation, both at Redmond and within the IT community. Microsoft was understandably concerned about its intellectual property rights, but IT pundits played up the security angle. Many unnamed (and some named) "security experts" were quoted as saying the leaks of the source code present a serious security issue, and that hackers could use the information to launch new and improved attacks against the Windows operating systems [10]. Does This Mean Open Source is Less Secure? These claims must seem confusing to those who have been listening to open source proponents, who for years have told us that their software is more secure precisely because the source code is readily available to everyone. If having the code "out there" makes Linux more secure, why would the same thing make Windows less secure? Of course, Microsoft has always taken the opposite stance. During the anti-trust trials, they argued vehemently against the court's proposed remedy of disclosing their source code based on the security risks of doing so. Who then is right?

Open Source Software Security

The availability of source code is a defining characteristic of open source software, but is not an unalloyed blessing [11]:

- Attackers can examine the source code for vulnerabilities to exploit. This can also be an issue with closed source software whose code is sometimes leaked, as was recently the case with the Microsoft Windows source code. There is a long history of leaks of source code and algorthims, include the COMP128 encryption algorithm used in GSM mobile phones.
- Defenders can examine the source code for vulnerabilities to fix. This commonly happens in one of two scenarios. The first—searching for a specific vulnerability that is being actively exploited—is relatively easy because the exploit can be reverse engineered and when the vulnerability



is fixed and the exploit stops working it is clear that the vulnerability has been fixed. The second—searching for vulnerabilities that are not being actively exploited—is a much harder and more open-ended task. Effort tends to focus on high-profile code with obvious security implications and these sections of code are normally secure, but a flaw in almost any part of the software can cause a vulnerability and finding these involves examining all of the source code, which may run to millions of lines of source code. Finding and fixing vulnerabilities is an effective way for open source contributors to gain the respect of their peers, so many contributors are motivated to look through sections of the code for vulnerabilities.

- Anyone can contribute to the source code. Another defining feature of open source software, the ability of anyone, anywhere to contribute source code is a double edged sword: if attackers can attempt to add vulnerabilities to the source code then defenders can contribute fixes to vulnerabilities—the process of finding and fixing vulnerabilities is not limited to the resources and office hours of a single company, but the resources of the users of the software. A result of this is that fixes for known or actively exploited vulnerabilities tend to be issued substantially faster for open source software than for closed source software. To prevent attackers from inserting malicious codes, most open source projects accept contributions only from trusted contributors [11]. All other contributors have to have their contributions approved by a trusted contributor. The process of becoming a trusted contributor varies, but often involves long-term commitment to a project and the exchange of cryptographic keys so that trusted contributors can communicate securely. The use of version control systems allows tracking of which contributor makes which changes, when and the reason they give. Should a contributor be found to have introduced a vulnerability, all of their contributions can be isolated, examined and purged.

Malware writers target large pools of uniform, poorly maintained computers.

Malware includes viruses, worms, trojans, spyware, rootkits and other software that has deliberate malicious effects or side effects. Currently open source applications and operating systems tend to be relatively diverse (one consequence of being able to modify the source code is that people do), relatively well maintained (because of the technical barriers to using much open source software it tends not to be the software of choice for complete beginners) and a relative minority (again because of the technical barriers); all these factors make it a poor target for the writers of malware.

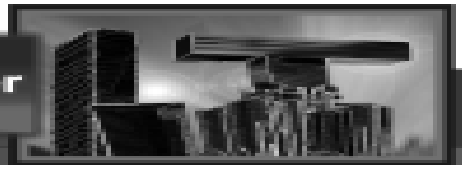
- The technical barriers to the use of open source are being lowered by better documentation and user interfaces, the diversity is being reduced through standardisation efforts and it is becoming widely used; these are progressively undermining the factors which make it unattractive to malware writers.
- The open source communities' acceptance of input from non-professionals and emphasis on skills development is likely to lead to an increasing number of individuals with the skills to write malware but without the professional ties which might prevent them from writing and releasing malware. This may lead to a long-term increase in malware.

There are so many versions of open source software that making sure a bug is fixed in all of them is impossible.

Anyone can make changes and or customisations of open source software and if a sufficient number do that it can lead to confusion over which bugs affect which versions [6].

- There can be a confusing number of versions of open source software, but generally distributors (who package open source software into usable systems) have tools that allow tracking of what software is installed and automated or semi-automated update of the software when security updates appear.
- The freedom to change also allows security specific customisation to be undertaken. Perhaps the best example of this is the Security Enhanced Linux project, undertaken by the USA's National Security Agency, which provides a significantly larger set of security features, at the price of significantly higher maintenance and operating overhead [9].
- Similar issues are faced by commercial companies who support several products. This is especially true of hardware vendors, for whom continuous upgrade to the current versions is not an option.
- The Common Vulnerabilities and Exposures system, is a vendor neutral system which provides a dictionary of vulnerability identifiers, effectively solves these problems for open source, closed source and mixed systems. [4]

These show that open source is not a magic bullet for security but can directly influence security trade-offs in complex ways.



Musings on open source security models -

Common sense would seem to indicate open source software is insecure because, for many, secure means hidden, secret. Recent discussions on several security-related mailing lists have revolved around keeping the names of server's secret, as if hiding information makes networks secure.

Others see open source as the means to secure operating systems. Some of the more secure systems available are based on the open source model. Many do not trust closed proprietary systems that cannot be examined and verified for secure coding. To them, it is a myth that such systems are somehow inherently insecure, even though this belief is widely held.

In cryptography circles, they have a saying: *The security of an algorithm should not depend on its secrecy* [12]. Now, this maxim is equally applicable to security software in general. Algorithms can be reverse-engineered. Protocols can be cracked through analysis. That which is hidden and secret will eventually be revealed. A secret, once lost, is gone forever and cannot be regained. Security through secrecy is largely a myth.

The argument then goes on, from the closed source camp, that secrecy or obscurity, when applied to an otherwise secure system, improves the security. It slows up intruders and, even when the secrecy is broken, the security remains as it would have been with open source.

So, all other things being equal, a secure system that is not open source should be more secure than a secure system that is open source. This sounds reasonable. Is it reasonable, though? Can all other things be equal? Are there ways in which secrecy and closed source code can actually *compromise* security?

The nature of common sense

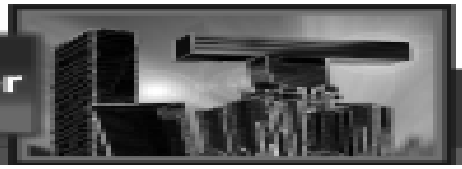
There are many opinions and definitions as to what comprises "common sense." By one definition, common sense is the "future application of past experience." This definition allows for the possibility that what some refer to as common sense may, in fact, be wrong. People unfamiliar with the open source model are accustomed to keeping their source secret. When their source does become public, it is almost always related to a security breach or the threat of a security breach. Revelation of code developed and maintained in secret also often results in the discovery of previously undetected flaws and security holes. It is no wonder, therefore, that those accustomed to the closed source model of development view open source as insecure. Their past experience with security breaches colors their conviction that security goes down the tubes when source code becomes public. It is in their "future application" of that "past experience" that common sense fails. Their past experience really no longer applies, since the conditions have changed.

The nature of secure software

Many people believe security is a functionality of software. But network security is a process, not a checklist on the side of a software box. Programmers of proprietary software leave holes and take more liberties than open source programmers [11]. The reasons are simple: Their management and marketing departments are screaming for the code to ship, doing it right is harder than doing it quickly and, after all, they think, who is going to know? [10] On the other hand, most open source software is written by people for whom programming is not a chore. It is a craft, and they take great pride in doing their work properly. Away from the demands of marketing and management, they are able to create the code that they want to write, not the code that will make the most money. The difference in the quality of the code produced by the two methods is staggering.

Proprietary software vendors claim that corporate reputations and the reputations of their developers are at stake with regard to security, but corporate reputations are easy to repair. After all, the "Love Letter" virus that exploited a security design blunder in Microsoft's email client was responsible for millions of dollars in lost productivity and lost data - an error that would have been avoidable if the code had been open to peer review. Yet Microsoft is still in business. With open source, developers' personal reputations are on the line. There is no corporate public relations spin machine to hide behind [1]. Also, consider the reasons for security alerts. Most often, a security alert is issued for a proprietary software package once a cracker has created and published an exploit to take advantage of a problem. Most open source security alerts are issued because of third-party audits, not published exploits, and an alert is published in the spirit of openness to notify any users of the broken software about upgrades.

Secure systems should not depend on the secrecy of the source. What is it, then, that makes a system secure? Secure systems require quality software utilizing secure coding techniques implemented and installed in a manner consistent with security guidelines and policy.



Myths regarding security and open source software

The following are some of the myths that contribute to the belief that open source is insecure:

Myth 1. There is no source control in open source software [5].

Those who develop open source software tend to howl with laughter over this one, but it is a criticism. Some people actually believe open source software is developed with total disregard for tracking, accountability, or control. But with large and diverse development teams being the norm in the open source world, source control is a *necessity,* not an option.

Myth 2. No one really looks at the source. [5]

The open source camp proclaims the source is available for anyone to examine. The closed source camp counters that people either do not have the time or the skill or would not invest the effort to examine it. Since the source is not examined by everyone or examined by them personally, the argument goes, it is as good as if it were examined by nobody, and any errors in the code are unlikely to be made public.

After the release of PGP 2.6, someone examining the code noticed an error in a random-number generator. The mistake was very minor. A statement that should have been an XOR with operation (~=) was in fact, an assignment (=). The result was that the random number seed was somewhat less random than expected. This didn't seriously compromise the security of PGP, but it did reduce the strength of the random keys. This error was quickly corrected, and the incident does illustrate some important points. It shows that the code is examined by others and that coding errors (intentional or unintentional) do get spotted. Due to the minor, obscure, nature of this buglette, it also indicates that the probability of a more serious bug or backdoor going undetected is rather low [10].

Myth 3. Anyone could put a backdoor or trapdoor in open source software [10, 11].

Open source uses source control, it uses code examination and analysis by others, and it puts the personal reputations of the authors on the line. No developer would personally risk his or her reputation by putting a backdoor in source that is openly available in public forums. This can be contrasted with closed source programs, which have an amazing array of "Easter eggs," those cute little surprises programmers leave in their code. What does the existence of such surprises say about the state of code review and source control in closed source circles? This begs the question: Are there backdoors and serious surprises we can not see? Easter eggs are cute and backdoors are not. outside of that, there is not much difference between them. Placing such secret surprises in open source would certainly seem much more difficult to do, if not a paradox in itself.

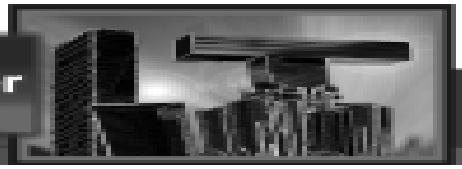
Myth 4. Hackers are going to find all the security holes in open source software [10].

Well, this is really not the myth. The real myth is that they would not find the security holes in closed source software. One only has to look at the security warnings and advisories attached to any of the closed source systems to realize this. It has become almost a joke in the industry that hackers (good and bad) probably have better debugging, analysis, and reverse-engineering tools than developers have. Unfortunately, this joke often ends up with a decidedly unfunny punch line for network administrators. The closed source camp likes to point out every open source security advisory as evidence that open source is insecure. In doing so, they conveniently ignore the counter examples in their own advisories. They also conveniently overlook the fact that open source problems are often found and fixed before they are widely exploited, while some closed source problem go un-addressed for months, or longer.

Not long ago, Alan Cox announced a Solaris security hole on the Bugtraq mailing list, after waiting over a year for Sun to fix the problem. Sun's response: that Alan had failed to notify the "correct people."[10] In contrast, the "Ping 'O Death" bug was fixed in Linux only a few hours after it was announced [9]. The same bug remained unsolved in some closed source systems for weeks or months. The author of the "teardrop" exploit only released its source after seeing David Miller commit the fix to the Linux source tree.

Who Will Be Responsible For Security Breaches?

Various licenses have been approved for licensing open source software. Perhaps the most popular and most widely recognized open source license is the GNU General Public License (the "GPL"). Under this license, there typically will be no warranties from the developer(s)[8]. If GPL software is designed with "back doors" or other malicious code that allows unauthorized access to information systems, who will be responsible? As one author has noted, "If nothing else, our tests



showed us that the security of open-source systems is not a cut-and-dried issue and that having someone to yell at usually is a good thing [3]. While commercial licenses also frequently provide limited warranties, those warranties typically are more extensive than open source warranties. In addition, for liability that cannot be disclaimed (such as for intentional misconduct), there are advantages to having a large commercial company to "yell at", rather than a dispersed collection of individual developers whose collective work may be the cause of a problem.

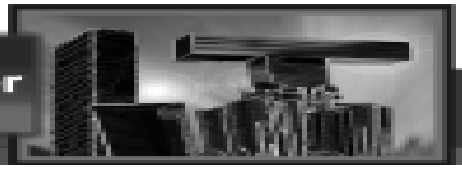
Conclusion

There are several reasons why open-source software provides for superior computer and network security, but the computing public seems confused about why this is so. The factor of openness, which holds that flaws are discovered and fixed because selfless programmers spend countless hours carefully combing through the source code and alerting the development teams is one reason, diligence is another. End users can place more confidence in their applications, utilities and clients when the source code is available for review by anyone who wishes to examine it. It is simply impossible to conceal spyware, adware and secret phone-home capabilities in products that can be examined freely. Open Source Software security however do not come from openness alone. They come in addition from the coincidence that open source systems, like Linux and BSD, are modeled on UNIX, which is designed in a modular fashion. Such systems are more transparent to the user or administrator, and have far fewer interdependencies - two factors that are exceptionally good for security.

Open source software has a long tradition in IT security - where reliability, flexibility, transparency and trust are vital. Security software must be reliable and without vulnerabilities like buffer overflows. The only economic way to look for such flaws is by checking the source code. That is exactly what some code audit projects are doing. They check the source code of application programs and even complete operating systems, line by line and eliminate security flaws. It is often said that as attackers have access to the open-source code, they are more able to attack a system. Attackers are often able to get the source code of operating systems, for example, because it was licensed to universities and is now available 'underground'. An attacker just needs to find one flaw in the software. This is possible without source code, for example, by feeding an application with random data. If the application crashes, a flaw exists and the attacker can look for ways to exploit it. Security software must be flexible. It is mostly off-the-shelf operating systems that are used for today's servers and firewalls. These operating systems are optimised for normal use in a friendly environment, but not for high-security demands. They need to be modified to reach an acceptable level of security and to survive in a hostile environment like the Internet.

References

1. ACE (2005). Does Open Source Software Enhance Security. *http://www.addict3d.org*. Downloaded February, 2005
2. Adeyemo, A. B. and Oladipo, O. F. (2004) Open Source and Proprietary Software: What's at Stake? Proceedings of the 19th National Conference and AGM of the Nigeria Computer Society ICADEV, 2004.
3. Chowdhry, P. *Open source meets the 'Baywatch' factor*, PC Week, pg. 83 (October 18, 1999).
4 *Common Vulnerabilities and Exposures* http://www.cve.mitre.org/
5. Deb S. (2004) Is Open Source Really More Secure? http://www.softpanorama.org/downloaded March 2005
6. ISO 9000 in brief from the International Organization for Standardization http://www.iso.ch/iso/en/iso9000-14000/index.html
7. Microsoft Press Release on the leaking of the Windows source code http://www.microsoft.com/presspass/press/2004/Feb04/02-12windowssource.asp
8. Oladipo, O. F. (2004). Building the foundation of an information driven society: the OSS/FS approach. National Convention of NACOSS. Ibadan, 2004
9. Security Enhanced Linux http://www.nsa.gov/selinux/index.cfm
10. SourceForge http://sourceforge.net/
11. Stuart Y.(2004). Open Source and Security. http://www.oss-watch.ac.uk Downloaded January, 2005
12. The RSA algorithm http://www.rsasecurity.com/rsalabs/node.asp?id=2146



Critical Mass of ICT in Africa: FOSS.NET Enabler

A. Kayode Adesemowo

ABSTRACT

E-Government implementation in Africa ought to be a panacea for good governance and effective service delivery for citizen. OSS adoption in e-government requires ICT skills capacity within the continent. The dearth of ICT personnel in Africa is not limited to FOSS but also proprietary platform. Africa human investment in ICT should be on open platform to develop pool of trained skills personnel, promotes vibrant localized researches whether on open or closed platform. The key for us is to start-off with bits building bytes, bytes forming packets streaming out African Initiatives and Technologies. Such skills would generally be transferable across platforms. .NET as ECMA/ISO standard provides a development medium for multi-platform, multi-language environment to develop ICT skilled personnel across Africa.

Introduction

National Governments especially in Africa are currently looking at or implementing e-Government. Such drive for e-government is expected to facilitate Government-to-Citizen (G2C) Service Delivery thereby promoting open governance. Another factor for e-government is balancing of service delivery against running cost (tax money, social grant). Thus, the ongoing need for operations optimization (process re-engineering, Business Intelligence/Solutions, converged ICT Infrastructure, skill update amongst others). One area that has been identified is the role of Open Source in curtailing ICT cost. Yet, without critical analysis, effective policy and structured strategy, the anticipated cost reduction becomes a myth. However, with operational optimization in place, Free and Open Source Software (FOSS) could be integrated into existing system without fear of pitfall.

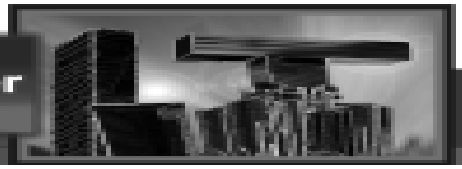
One critical element normally glossed over in adopting FOSS is ICT capacity to implement and maintain the FOSS system. FOSS tends to normally offer lower initial cost or lower barrier for SMME (little or no Licensing cost). Yet, it incurs operational support expenses compounded by expatriate FOSS skilled personnel.

The dearth of ICT personnel in Africa is not limited to FOSS but also proprietary platform. Africa human investment in ICT needs good governance to be able to trigger forth on its own terms as future giants and leap frog the digital divide. With critical mass of ICT personnel, Africa will be overwhelm with large supply of computing resources to developed African solutions. Wherein lies Capacity Building? It is in building a critical mass of developers. We need pool of trained skills personnel, vibrant localized researches whether on open or closed platform. The key for us is to start-off with bits of 1's building Bytes, continuing with bytes to form packets and enough packets to stream out African Initiatives and Technologies.

Educational Literacy of computing knowledge should be followed on Open Platform rather than forced to particular proprietary or FOSS platform. Such skills would generally be transferable across platforms.

A case-study of open-standard as bases of developer bit-by-bit is the .NET platform as an ECMA standard that provide a Common Language Runtime (CLR) environment to program in any compliant language of choice. C# has been ratified as ECMA-334 and available as ISO/IEC 23270:2003 free standard with .NET as ECMA 335 and ISO/IEC 23271:2003. .NET has been deployed as Proprietary and 'free' (.NET Framework), on another front as OSS (Mono) and Free (Portable.NET). It has been embedded into Integrated Development Environments (IDE): Microsoft's VS.NET, OSS SharpDevelop and Free Software's MonoDevelop. More so, WebMatrix is a 'free' Web tools. Many attempts have been made to provide APIs that are exposed to multiple languages, cross-language integration and interface-based programming across varied platform. .NET has provided developers and researchers alike, with API and source access, which enables contribution to development of better technologies. Aside the availability of the .NET to developers across multiple platform, it is also an enabler for vibrant community one of which is the SADeveloper.NET in South Africa.

Effective Policy on Open Standard in Africa should help with capacity building toward enabling ICT as a medium for development.



e-Goverment

Governance in the 21st century extends beyond citizen rule to citizens upliftment and advancement within the limit of law. National Governments especially in Africa are currently looking at or implementing e-Government [7]. Such drive for e-government is expected to facilitate Government-to-Citizen (G2C) Service Delivery thereby promoting open governance. Another factor for e-government is balancing service delivery against running cost (tax money, social grant...). E-government has therefore bring on the horizon a new way of rendering services to the populace. Inner working of government are exposed as interfaces for citizen to interact with effectively, beyond geographical boundary. E-government nevertheless brings with it challenges and FUD (Fear, Uncertainty and Doubts) in its implementation and adoptions. It is hinged on thought through internal process analysis, re-engineering, re-designing and deployment. More so, services are not to be centered on department services but on addressing citizens need.

e-Government could be define as the continuous optimisation of government service delivery, constituency participation, and governance by transforming internal and external relationships through the optimal utilisation of appropriate Information and Communication Technologies (ICTs). Thus, the ongoing need for operations optimization (process re-engineering, Business Intelligence/Solutions, converged ICT Infrastructure, skill update amongst others).

ICT for Development

It is essential for ICT not to be seen, viewed or adopted as a technological tool, rather it be a medium for development [12]. ICT for Development (ICTD) simply put is employing and aligning ICT as a medium of development rather than just a technological tool. Essentials of ICTD includes enhance service delivery, cost optimization, operational efficiency, real access and social development. Generally, ICTD refers to the external where it acts as catalyst and an agent of change for bringing improved service delivery resulting in higher standard of living.

There are three components to e-government: e-administration, e-services and e-society [7].

- **e-Administration - Improving Govt. Processes**

  E-Government initiatives within this domain deal particularly with improving the internal workings of the public sector. It involves cutting process costs, managing process performance, making strategic connections in government, creating empowerment

- **e-Citizens and e-Services - Connecting citizens**

  Such initiatives deal particularly with the relationship between government and citizens: either as voters/stakeholders from whom the public sector should derive its legitimacy, or as customers who consume public services. Government interactions includes talking to citizens, listening to citizens, and improving public services

- **e-Society - Building external interactions**

  Such initiatives deal particularly with the relationship between government agencies and other institutions such as public agencies, private sector companies, non-profit and community organisations. It encompasses working better with business, developing communities, and building partnerships.

It is imperative therefore, that e-government should be properly aligned with ICTD for meaningful engagement and development. From the mapping in Table 1, ICTD is the enabler for e-government. It transcend beyond mere technological tool.

The mapping between ICTD and e-government shows their dependency and coupling.

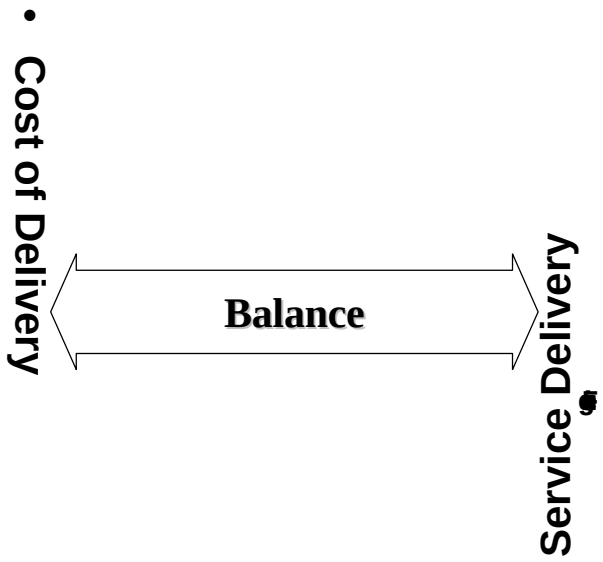
ICTD is the process for improved service delivery by enhancing productivity and efficiency. It is also a catalyst and enabler for citizen Real Access medium to strengthening good governance and broadening public participation as well as improving the quality of life for all, especially disadvantaged communities.

Government internal ICT business solutions requires critical analysis, re-engineering and re-positioning for

- Essentials of ICTD:	- **Improving govt. processes: e-Administration**
- Cost optimization - Operational efficiency - Enhance service delivery Real access and social development	- Connecting citizens: e-Citizens and e-Services - Building eexternal iinteractions: e-Society

Table 1: Loose mapping of ICTD with e-government

it to have the capacity to being an agent of change for e-government and e-governance. There is a need for balancing out cost of delivery to service delivery.



1: Balancing Cost of Delivery with Service Delivery

In enhancing service delivery, the cost of delivery has to be optimized to balance and curtailed public expenditure.

Internal ICT business solution optimization comes at great cost especially when not properly re-engineered. This is so when it is product driven rather than solution driven. Underlying technologies within solutions must be properly identified, analysed and ensure it align to enterprise solution.

One area that has been identified is the role of Open Source in curtailing ICT cost [4],[6]. Yet, without critical analysis, effective policy and structured strategy, the anticipated cost reduction and benefits becomes a myth. However, with operational optimization in place, Free and Open Source Software (FOSS) could be integrated into existing system without fear of pitfall.

FOSS

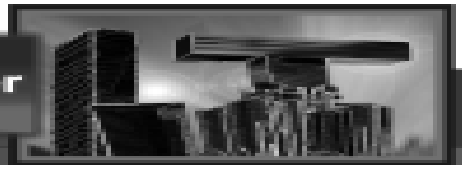
Free and Open Source Software (FOSS) is a general term use to classify Software and solutions conforming to both or either of Free Software and Open Source Software (OSS). Prior to the emergence of OSS, sourceware has loosely implied OSS or FOSS. Larry McVoy relates sourceware in this loose context [9]. FOSS within the context of adoption ought to refer to underlying or bundled Licensing rather than implied or labelled License. GNU General Public License (GNU GPL) is the root license (http://www.fsf.org/licensing/licenses/gpl.html). Other licensing as allowed by Open Source Initiative must align or be compatible with GNU GPL. "Free software" is a matter of liberty, not price. Free software guarantees users' four defined freedom (to run, copy, distribute, study, change and improve the software. It is hinged on the *Copyleft* rights and GNU GPL Licensing [5].

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

Open Source Software (OSS) is a derivative of Free Software without the stringent moral freedoms rights. OSS is made up of many licensing, which are defined by the Open Source Initiative [11].

In many enterprises, FOSS co-exist with other solution. This is categorised as OSS hybrid, which refers to an environment that uses a FOSS application or solution on a proprietary operating system or vice versa. Other emerging initiatives are the shared source solutions where proprietary software source code access are given to relevant parties or partners on request and shared within the (defined) community. Microsoft Shared Source Initiative is an implementation [10].

FOSS implementation should promote efficiency and cost effectiveness. Thus, it must be understood that FOSS is not a PRODUCT, rather it is a solution platform whose adoption enable Cost of Delivery optimization. FOSS is not a panacea to solving ICT problems; rather it is an enabler for process optimization and people skill culture. Open standard [8] is still essential and vital in process optimization to which FOSS can and should contribute to. Without a focus on open standard, FOSS adoption would not bring the optimized benefits anticipated and might just become another cloak in the enterprise process. Building skill on open platform is essential to building a culture within the enterprise. This is noted in [9] with Unix stratification and a move to closed platform. The essential of mass skill culture is discussed under ICT Critical mass section.



The NET Standard

In order to relate the essence of open standard in building critical skill mass, this paper has identified the .NET standard and its implementations as a base for developer bit-by-bit

The .NET platform as an ECMA standard provides a Common Language Runtime (CLR) environment to program in any compliant language of choice. C# in addition to the CLR have been ratified as ECMA-334 & ECMA 335 respectively [1],[3].

C# and .NET CLR in this paper do not refers to Microsoft Corporation commercial .NET framework or products. The Standard ECMA-335 defines the Common Language Infrastructure (CLI) in which applications written in multiple high level languages may be executed in different system environments. Thus, there is no need rewriting the application to take into consideration the unique characteristics of those environments. CLI is a runtime environment, with the following

- a file format;
- a common type system;
- an extensible metadata system;
- an intermediate language;
- access to the underlying platform;
- a factored base class library.

The ECMA 334 Standard specifies the form and establishes the interpretation of programs written in the C# programming language. It specifies

- The representation of C# programs;
- The syntax and constraints of the C# language;
- The semantic rules for interpreting C# programs;
- The restrictions and limits imposed by a conforming implementation of C#.

As a standard [8], .NET is free for implementation. In its implementation, add-ons and extensions have been made to make it robust. The major successful implementation is no doubt that by Microsoft (MS) with its .Net framework and Visual Studio development IDE. It is of note that the commercial offering MS .NET as publicly known is proprietary. A streamlined shared source alternative geared towards academia and research is available as Shared Source CLI known as rotor (http://msdn.microsoft.com/net/sscli). Rotor is released to run on Windows, FreeBSD and MacOS. Work done at Rhodes University, South Africa has extended it to the Linux platform. Within the space of rotor release, Miguel de Icaza of Ximian and GNOME spearhead the Mono project (http://www.mono-project.com). Mono is an implementation of the .NET standard with the goal of porting MS.NET to the Linux platform. The project has been a success with its own extensions such as the GTK#. Novell bought over Ximian in 2004 and Miguel de Icaza has joined Novell still coordinating Mono along with his role as vice president of product technology. The free software implementation is the DotGNU Portable.NET meta-project managed by Rhys Weatherly (http://www.southern-storm.com.au/portable_net.html). On the embedded platform, main implementation is the Microsoft Compact framework, which has been extended with the open source OpenNETCF (www.OpenNETCF.org).

Development tools availability are on the increase. Most advance presently is MS VS.NET, however the sharpdevelop is catching up fast and is robust for most development. Mono's monodevelop is being rapidly improved on. Omnicore X-develop is a new multi-language cross-platform IDE for .NET, Mono and the Java platform on Windows and Linux. It offers sophisticated semantic-driven productivity features for C#, Java, JSP, J# and Visual Basic.NET. .NET is also being made available for the eclipse platform as well. WebMatrix is a 'free' Web tools for .NET (http://www.asp.net/Default.aspx?tabindex=0&tabid=1).

There are a number of project on sourceforge (http://sourceforge.net/projects) built on and for .NET. A number are tools to develop effectively in .NET platform such as Ndoc, an extensible code documentation generation tool for .NET developers. Others are listed:

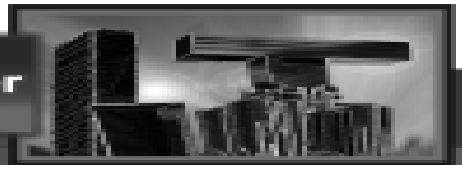
NxBRE, a lightweight Business Rule Engine (aka Rule Based Engine) for the .NET platform, composed of a forward-chaining inference engine and an XML-driven flow control engine. It supports RuleML 0.86 Naf Datalog and Visio 2003 modeling.

Netron - Generic diagramming, graph-drawing and graph-layout kit for the Microsoft .Net framework

DotNetNuke - An open source Web Portal Framework / Content Management System application written in ASP.NET / VB.NET for the Windows OS platform.

CruiseControl.NET - Automated Integration server , code integration on code checkin

IIOP.NET allows a seamless interoperation between .NET, CORBA, and J2EE distributed objects. This is done by incorporating CORBA/IIOP support into .NET, leveraging the remoting framework. http://sourceforge.net/projects/asp-ent-man/



Many attempts have been made to provide APIs that are exposed to multiple languages, cross-language integration and interface-based programming across varied platform. .NET has provided developers and researchers alike, with API and source access, which enables contribution to development of better technologies. Aside the availability of the .NET to developers across multiple platform, it is also an enabler for vibrant community one of which is the SADeveloper.NET in South Africa.

FOSS.NET

A number of the project listed above shows the community thriving on .NET across multiple platforms. The growth of .NET is also hinged on the availability and continuous expansion of communities proving forum for support on .NET. The gotdotnet is a forum and community that focus on .NET technologies providing tools, training, samples, products, article and tutorial (http://www.gotdotnet.com). The codeproject is a showcasing of projects on .NET technologies. Gotdotnet, however provides a project hosting facility as well much as sourceforge.

In South Africa, the SADeveloper.NET is a community of developer meeting to discuss live project issues across platform though with more emphases on .NET. The online forum provides a meeting place for discussion in addition to the monthly face-to-face meeting and other events. It is a great ground for Networking with fellow developers as it provides social space and place. As with any open source community, source codes, information, solution, material are freely shared. Tutorial and training are sometimes organized free. Such active community has a role to play in building network of developers and localizing projects in Africa. One element of the SADeveloper is the active encouragement for and participation by student within its locale. This is essential in building the bond necessary for grooming and capacity building.

An evolving community for students geared toward .NET technology is the projectfirefly forum (www.projectfirefly.co.za). While the project is an annual competition showcasing students projects built on .NET technology, a community is gradually forming round it. It has an online forum presently where technical and non-technical discussion alike takes place.

ICT Critical mass

FOSS.NET provides a platform to build skill and developed solutions across multiple platforms and great potential to honed skills in language of choice. Such portability of skill across domain is crucial in getting ICT personnel on speed without the need for humongous retraining and retooling. One factor for the wide adoption of Java is the portability of skill across platform. Yet .NET offer greater flexibility with its multi-language feature if and when properly harness. This will be a strong factor in developing ICT skills without lock-in to a particular platform or OS.

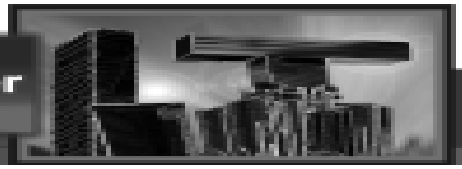
Larry McVoy in his concern on fixing Unix proliferation comment on the hackers interest, "the hacker community, which includes universities, research laboratories, and many people at private corporations, is interested in the best, most widely used, free software they can get". He went further to alert the danger in business entity not recognising the importance of critical mass of developer (hackers) saying, "... to dismiss the hackers as 'nuts' is a disaster" ... in fact "to dismiss the hackers is to dismiss products that produce financial reward". Thus, he proffer that we should "... allow business world and the hacker world to coexist and benefit one another." Larry's conclusion is the essence to move towards sourceware, which is now known as FOSS.

FOSS brings to the table a leveling platform for development and skill acquisition. Its benefit is numerous for individuals, small and medium enterprises. It offers no licensing cost, freedom, access to source code, right to modify and redistribute [2]. These obvious advantages nonetheless still require implementable APIs to program to, explicit documentation to refer to, local community to relate with and standards to adhere to. Bringing all elements together, it could be seen that FOSS.NET provide suitable platform to developing skill irrespective of platform (Windows, Linux, Mac) or model (FOSS or proprietary).

In theory, it is 'once develop all deploy'. The challenges (of maturity, extensions) of now nonetheless, prove of concept has been laid, while implementations are maturing and development tools with IDEs advancing.

One critical element normally glossed over in adopting FOSS is ICT capacity to implement and maintain the FOSS system. FOSS tends to normally offer lower initial cost or lower barrier for SMME (little or no Licensing cost). Yet, it incurs operational support expenses compounded by expatriate FOSS skilled personnel.

The dearth of ICT personnel in Africa is not limited to FOSS but also proprietary platform. Africa human investment in ICT needs good governance to be able to trigger forth on its own terms as future giants and leap frog the digital divide. With critical mass of ICT personnel, Africa will be overwhelm with large supply of computing resources to developed African solutions.



Herein lies Capacity Building. In building a critical mass of developers, we need pool of trained skills personnel, vibrant localized researches whether on open or closed platform. The key for us is to start-off with bits of 1's building Bytes, continuing with bytes to form packets and enough packets to stream out African Initiatives and Technologies.

In analogy to the OSI layer as shown in Figure 2, at the physical layer, the pdu (protocol data unit) is simply bits. Bits are simply 1's and 0's at this layer and on the medium. They are independent of types of network whether Ethernet, frame relay or any other.

The 1's and 0's are grouped together as nipples or byte's as frames. This is where the synergies start. Whatever the medium (OS platforms), skill acquisition could be impacted with .NET irrespective of high or low end PC. Different bits of skills scale up as skills acquisition progress up and specialist starts emerging at the top of the stack. The application layer at the top of the OSI layer provides services to applications running on the system. In similitude, skills build up on .NET framework can be applied to wide and varying needs on the continent. Integrations could be done with other type of technologies. It is of note that skills must be developed on this other technologies and platform as well.

OSI layer - Networking		FOSS.NET .NET Technology
Application		Specialist skills/Hackers
Presentation		Integration
Session		Synergy Coordination
Transport		Multi Domain
Network		Competencies
Data Link		Bytes - Bonding
Physical		Bit by bit – Skill Isolation

Figure 2: FOSS.NET in OSI concept

The OSI layer is use in analogy to FOSS.NET bit-by-bit skill acquisition in developing critical mass of developer for Africa

.NET is not being touted as the only platform to developed skills in Africa. Nay. Rather, it is a showcase of how to quickly get developers build up across platforms. It therefore gives leverage for acquired and donated PC and Labs to be optimized without lock-in to platform and OS religious war.

Conclusion

Educational Literacy of computing knowledge should be followed on Open Platform rather than forced to particular proprietary or FOSS platform. Such skills would generally be transferable across platforms. In Africa where donated PC is a reality, multi-platform enabling technologies such as .NET ensure optimum leveraging of such PC for skill acquisition.

Effective Policy on Open Standard [8] in Africa should help with capacity building toward enabling ICT as a medium for development.

Africa should pursue skill acquisition for African on open standard with tolerance for de-facto standard, rather than push or maneuvering to particular platform whether Windows or Linux.

With critical mass of developers in place, requisite platform is available to follow lines of Germany, India, China, Brazil and the like to push a particular specific platform of choice. Without a critical mass of developer, Africa will find herself enslave in the knowledge economy with expatriate hackers and specialist.

References

[1] .NET CLR ECMA standard. Available at http://www.ecma-international.org/publications/standards/Ecma-335.htm

[2] Bakker B., "Shrugging off the licensing yoke", ITWeb Brianstorm - Third special focus on Linux, pp. 23. Dec. 04/ Jan. 05

[3] C# ECMA standard. Available at http://www.ecma.ch/ecma1/STAND/ECMA-334.htm

[4] CeI FOSS Strategic Framework Document, Center for e-Innovation, Provincial Government of the Western Cape, South Africa.

[5] Free Software Foundation, "The Free Software Definition". Available: http://www.fsf.org/philosophy/free-sw.html

[6] Government OSS Strategy Framework Document, South Africa Government Information Technology Officers' Council (GITOC). Available at http://www.oss.gov.za/modules.php?op=modload&name=Downloads&file=index&req=getit&lid=6

[7] Heeks R., "e-Government in Africa: Promise and practice", Information Polity 7, pp. 97-114, (2002)

[8] Heintzman D., "An introduction to open computing, open standards, and open source". Available: http://www-106.ibm.com/developerworks/rational/library/1303.html

[9] Larry McVoy, "The Sourceware Operating System Proposal Rev 1.8", 9 Nov 1993. Available online at http://www.redhat.com/support/wpapers/community/freeunix/freeos.pdf

[10] Microsoft SSI, Basic Principles of Software Source Code Licensing, http://www.microsoft.com/resources/sharedsource/Articles/LicensingBasics.mspx

[11] Open Source Initiative, "Open Source Definition", Available: http://www.opensource.org/docs/definition.php

[12] UNDP, "Essentials: Information Communications Technology for Development", 5 September 2001

Acknowledgement

Speaker biographical information

A. Kayode Adesemowo is a Technologist (Policy and Strategy) with the Center for e-Innovation, Western Cape Provincial Government, South Africa. His interest includes Proprietary and FOSS analysis in developing critical mass of ICT personnel in Africa. Others are mobile computing, wireless technologies, SIP/SIMPLE based VoIP solutions as tools for Africa developments. He is presently completing a research Masters (thesis) in Computer Science at the University of the Western Cape. He is a member of the Institution of Electrical Engineers, UK and Computer Society of South Africa.

The Open Source Alternative

Adewunmi Fagbemi

ABSTRACT

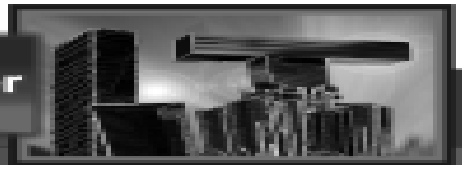
Open Source software is software licensed under terms that give its users three freedoms. The Right to: make copies of the software and to distribute those copies access the software's source code, and make and distribute improved versions of the software. The Internet is built atop open source software and open standards, and many of the open sour ce products are market leaders e.g. LAMP – Linux, Apache, MySQL, and Perl . The idea behind Open Source is when programmers can read, redistribute and modify the source for a piece of software, it evolves faster - improved, adapted, fixed and redistributed. Open Source is usually but not always free likewise, not all free software is Open Source. The need to embrace Open Source software in Nigeria cannot be overlooked. This need was not apparent earlier due to availability of pirated software. This is coming to a quick end with the clamp down on pirated software-retailers with the efforts of NCC, Microsoft and other proprietary software developers. This would invariably show the true picture of ICT in Nigeria as proprietary software when legally purchased and licensed is beyond the reach of most home users and small businesses. The cost of a thin client system built on proprietary software and that built entirely on open source software are not comparable. The adoption of Open Source as an alternative can be absolute, in that, the underlying operating system OS is Open Source or it can be partial, where the underlying OS is Closed Source but other applications are Open Source. In an absolute adoption of open source, there are many open source OS, Linux being most popular, exists in a variety of flavors. These different flavors of Linux ship with full software packages such as office suite, accounting, multimedia, graphic tools, scientific and mathematical applications, Internet and networking tools, programming suites and games. Some of these software packages have been ported to Windows which invariably facilitate a partial adoption of Open Source. Open Source can not only be used freely, it can also be modified to suit the user's peculiar needs. There has been the slow growth of open source awareness in Nigeria, and more users are beginning to experiment with Open Source while more corporate bodies are partially adopting Open Source in their production systems. The greatest fear to acceptance of Open Source is Security. It is a common misconception that Open Source is less secure than closed source software. This is not necessarily the case. In 2000, LinuxVirtualServer Enterprise Clustering Package was shipped with a backdoor. This backdoor was discovered about a week later. This quick discovery and fix would have been impossible without the codes being freely available. It would have taken longer to discover in closed source software, likely after an exploit. With appropriate collaboration and sponsorship, Open Source driven projects / initiatives can be developed. And in time , have Government ICT policies that would promote Open Source development, hence a technology renaissance in Nigeria.

An Insight into Web-Based Application

Adewole A., Philip, Sofoluwe A. B.

ABSTRACT

This paper focuses on the emerging area of web-based application and presents an overview of the opportunities and challenges in this area. The paper begins with an introduction of the World Wide Web and Web-Based software applications. Next, various basic tools for Web-Based application, how Web-Based applications work and an overview of the decisions that a developer must make when designing and implementing a web-based application are discussed. The paper concludes with practical examples of Web-Based application.

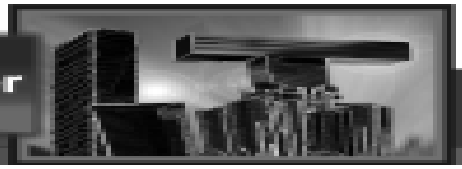


The IT Industry and the Academia in Nigeria

Z. Lipcsey; E. E. Williams, R. C. Okoro; E. O. Ukem

ABSTRACT

Information Technology (IT) deals with creation, dissemination and distribution of information. Academia is the field of learning, research and teaching, especially at higher institutions. Activities of the academia all require information, which, of course, is mostly the product of IT. The aim of this paper is to show how IT industry and the academia can be of mutual benefit to each other. The interaction between IT industry and academia is a complex issue, since it involves one using the products of the other (graduates are employed by industry). On the other hand, the accumulated knowledge of the academia is used by the industry and both contribute to each other's development. The paper attempts to analyze the situation and calls attention to the need for some form of co-operation, which could have far reaching positive impact on both parties. The fact remains that the IT industry requires the academia and vice versa. The IT industry from this work is supposed to depend mostly on the request for data/information by the academia. In other words, the academia is expected to fashion or shape the IT industry. The paper also considers this situation and advocates for much more collaboration between the industry and the academia.



Foss and Short Message Service in our Educational System

Oguntuase, Olufemi Ayobami

ABSTRACT

The advent of Mobile telecommunication in the Nigeria has made a profound impact in the society. Several areas of the society which has been impacted include the Educational system, Banking Industry and Communication System. Furthermore the need to send short message service through mobile phones is increasingly becoming important due to the high speed of delivery to the recipient regardless of where the recipient is located as long as the recipient is within the network of the GSM provider. Short Message Service (SMS) is the transmission of short text messages to and from a mobile phone, and/or Internet Protocol address. SMS is a messaging system which is provided by Global System for Mobile Communication Provider. Currently SMS is being supported by GSM, CDMA, TDMA mobile phone networks. Although services based on SMS have been feasible for many years, the recent mobile phone penetration and large scale adoption of the existing services by users, have made the SMS based services even more attractive to service providers. Free and Open Software is a type of software which the user is free to run, copy, distribute, study, change and improve the software. In other words the software can be freely redistributed, source code assessable to users, must not discriminate against any person or group of persons, must not restrict anyone from making use of the program in a specific field of endeavor, must not place restrictions on other software that is distributed along with the licensed software, allow users modifications and derived works. To this extent, the paper shows the importance of the Short Message Service to the Educational System, It also reveal the method that can be applied in the development of SMS using free and open source software tools.

INFORMATION TECHNOLOGY CAPACITY BUILDING: Impacting the Future of Nigeria's Economic Growth through Software Engineering Capacity Building and Management Practices

Dungor, Blessing Ngozi and Oladumoye E. F.

ABSTRACT

The demand for Information Technology (IT) products in Nigeria particularly, the software engineering, is on the increase and IT itself is currently influencing all aspect of economic life that controls the functioning of our present modern society. Access to IT products is also becoming increasingly essential to all those things associated with quality of life, such as economic development, eructation, healthcare, business, public services etc. In order for IT products to be beneficial, most people should be able to use them effectively and efficiently. It is very pertinent to note that most places and people in Nigeria do not have access to basic and efficient IT product either hardware or software. This paper discussed the following:

- The dynamic nature of Software engineering and its management in Nigeria
- The need for continuous capacity building of all the necessary software resources (Man power Infrastructures, materials, intellect, etc) in order to support the Federal government in its efforts to provide a critical IT software needed to leap-frog the nation into the information age, thereby, enhancing her e-readiness and elevating Nigerian economy from its present stage in the global economy as regards the software engineering Industry.
- Capacity Building needs for the Nation
- Strategic planning for Software Engineering
- Effective methods to adopt for effective capacity building
- Different levels of Software training (Locally or internationally) which will eventually help build up Nigeria's Indigenous Software Engineering industries by substituting the imported software with totally indigenous one
- Provision of training that is affordable by individuals who will like to acquire such knowledge
- Problems being faced and the envisaged ones will also be discussed.

8th International Conference

ISPAN

INSTITUTE OF SOFTWARE PRACTITIONERS OF NIGERIA

ITAN

nigeria computer society

NITPA

ISPAN
NACOSS

nigeria computer society

www.ncs-nig.org