GLOBAL JOURNAL OF MATHEMATICAL SCIENNCES VOL. 8, N0. 1, 2009: 49 - 63
 ISSN 1596-6208

AUTOMATING THE MANAGEMENT OF SOFTWARE PROJECTS IN A DEVELOPING IT ECONOMY

I. I. ARIKPO and A. O. OSOFISAN

(Received 5 November 2008; Revision Accepted 4 March 2009)

ABSTRACT

Software project management is the control of the transformation of users' requirements and resources into a successful software result (product). This work automates the management of software projects in an emerging IT economy like Nigeria. It also explores the simulation of management practices such as configuration management and risk management. The COCOMO II model was employed for the estimation process, while the Risk Model from The American Systems Corporation (ASC) was used for risk management. Experimental data was obtained from AcadSoft Solutions, Calabar, Unical Computer Centre, and OmegaBiz.ng Software Solutions & Consultancy, Calabar. The resultant network-based software tool was developed on object-oriented technology using Java. The study established that good management practices may still be applied by the Nigerian software industry that lacks expertise in software management. Multi-site development approach facilitates large projects by using simple network-based application that aids collaboration among team members. Future research could extend to system and real-time software projects, to give a holistic picture of software project management in developing countries.

KEYWORDS: Software project management, configuration management, risk management, multi-site development.

1.0 INTRODUCTION

Software Project Management can be defined as the responsibility for producing desired software with acceptable limits of resource usage. It encompasses the knowledge, techniques and tools to manage the development of software products (Tomayko 1989, p.4).

In his book on Software Project Management, Page-Jones (1985, p.1) made the statement:

> "I've visited dozens of commercial shops, both good and bad, and I've observed scores of Data Processing managers, again both good and bad. Too often I've watched in horror as these managers futilely struggled through nightmarish projects, squirmed under impossible deadlines, or delivered systems that outraged their users and went on to devour huge chunks of maintenance time."

What was described above is a barrage of symptoms that result from a chain of management and technical problems. Hence, if a post mortem were to be conducted for every software project, a likely consistent theme will be encountered: project management was weak.

The management of software projects has led to software development failures and successes. As the software industry keeps growing rapidly, software engineers are struggling hard to grapple with the ever-increasing complexity of software development. Even as software engineering remains a people-intensive process, software technologies, processes and methods have advanced appreciably (now we have Computer-Aided Software Engineering, CASE), and therefore, techniques for managing people, technology, resources and risks in software projects, have profound advantage (Page-Jones 1985).

One of the fundamental problems affecting software industries in both developed and developing countries is poor project management. Poor project management practices is the

I. I. Arikpo, Department of Mathematics/Statistics & Computer Science, University of Calabar – Nigeria
A. O. Osofisan, Department of Computer Science, University of Ibadan – Nigeria.

major cause of delays in software projects, overspending of IT budgets, and chronic problems with dependability – safety, reliability and security – and maintainability of software products. The worst hit are the developing countries. This makes it difficult even for indigenous software consumers (like banks, educational institutions, government ministries, companies and parastatals) in these developing countries to depend on local software industries for their software needs. One can imagine the consequences of a fatal program crash in a banking software (with customers waiting) that may require complete dismantling of the software architecture!!!

To develop high-quality, globally-competitive, dependable and maintainable software products, software industries in developing countries need to rely on modern engineering practices. Such practices comprise a variety of methods, tools and techniques that are based on sound and good software project management framework. Experience has shown that such frameworks are closely tied to the more than five decades of computer system evolution. According to Naisbitt (1982), the transformation from an industrial society to an "information society" has profound impact on our lives; and information and knowledge – *controlled by computers* – has become the focal point for power in the 21st century (Feigenbaum & McCorduck 1983). This work designates four eras of software evolution, as schematized in Figure 1 below:

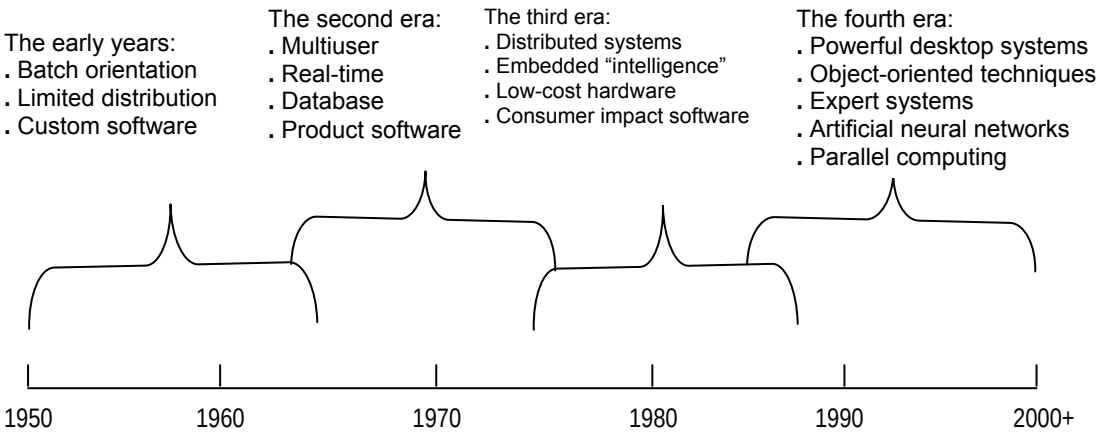

Figure 1: Evolution of software

Some Developing Countries such as India, has consolidated its software industry (example, Bangalore), while countries like Nigeria are just struggling to locate and position their software industry compasses. This is partly because, in Nigeria and some other developing countries, software development is undertaken mostly by non-professionals – people who just see software development as a program-writing activity and not a management and development task – because there are few software engineers available; while in India, software development is left in the hands of professionals, as India turns out approximately 290,000 engineering graduates (including software engineers) annually from her universities across the country (Radhakrishnan 2004, p.2). Besides, New Delhi-based NIIT alone has graduated over 200,000 Indians from its programming courses (Gibbs 1994). This is a major source of strength for India's software industry and accounts for over $5 billion worth of her software export services (Jalote 2000).

However, in the last five decades, software systems have evolved into a more complex routine. Some systems have over 50,000 lines of high-level language code, hence, the management of software systems often lead to "software crisis" (Gibbs 1994). Thus, the need arises to develop procedures that can handle, among others, the development processes and maintenance of software.

2.0 OBJECTIVE AND SCOPE OF RESEARCH

The purpose of this work is to come out with a software tool that will encourage a disciplined and management approach to software development in a developing IT economy like Nigeria, taking into consideration such infrastructural constraints as low technical manpower. The

web-based tool will also facilitate team participation and management in software projects, as against the “one-man” project approach commonly practiced in most developing countries.

As software is complex and dynamic, so is its management. Software project management is not very straightforward. Every software project has its own peculiarity, and unnecessary generalizations can be misleading.

This research work is therefore not exhaustive. Each developing country has different constraints, which in turn impact differently on the management of software projects. Besides, within the developing IT economies, some countries (e.g., India, Malaysia, Singapore, etc) are by far ahead of others (e.g., Nigeria, Ethiopia, etc.), which makes it difficult to develop a generic project management tool that will fit into, and address all project management problems in all developing countries.

In this study, the researchers are more concerned with countries like Nigeria, which are just emerging and trying to evolve their respective software industries. The resultant project management tool is targeted at these countries. This does not suggest complete automation of all the facets of software project management of countries in this category. Some basic knowledge of software projects is assumed, because, there are many aspects of project management that require human judgment.

3.0 STATEMENT OF THE PROBLEM

Software is expensive to develop and it is a major cost factor in corporate information systems’ budgets. With the variability of software characteristics and the continual emergence of new technologies, it is becoming more difficult to correctly estimate software development costs. For products developed for mass markets, such as Microsoft Office, the cost of development is not visible in the price of the product. However, for bespoke (custom) development, the software is targeted for one or a small number of customers, and therefore development cost influences the price. It is of strategic importance for an organization, whether as a customer or a developer, to be able to base its purchase or sales decisions on the ability to estimate the cost of development correctly and consistently (Page-Jones 1985).

Effective software project estimation is one of the most challenging and important activities in software development. Proper project planning and control is not possible without a sound and reliable estimation. As a whole, the software industry does not estimate projects well, and does not use estimates appropriately (Page-Jones 1985). From the point of view of the researchers, “we suffer more than we should, and as a result we need to focus some effort on improving the situation”.

4.0 RESEARCH METHODOLOGY

The methodology adopted in this work is to conduct an extensive study of software industries of some developing countries, including Nigeria based on the basic activities in software project estimation as shown in Figure 2 below. While for Nigeria, a survey of some software organizations was done and an observational methodology sometimes adopted, where the need arose; the researchers depended on published literature on software strategies of other developing countries (Tessler & Barr 1997). Besides, a comprehensive study was conducted on estimation models and other software management components (e.g. risk management) in developing countries vis-à-vis infrastructural constraints.

An implementation-driven methodology was then employed to demonstrate the software tool resulting from this research, using some sample software projects.

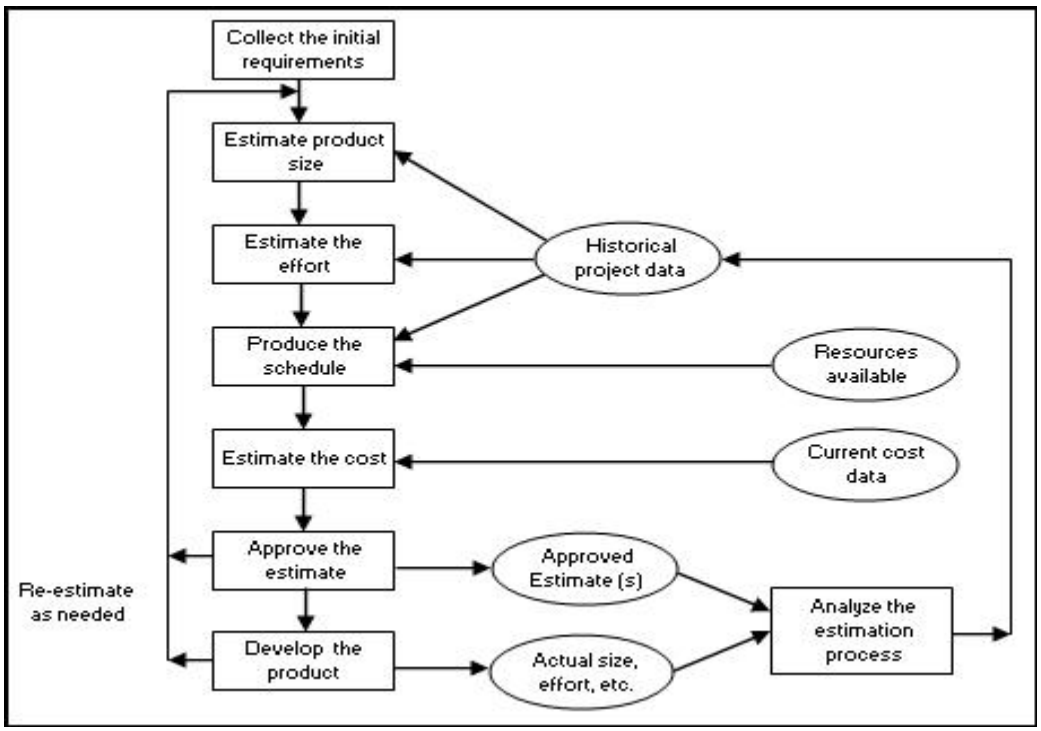

Figure 2: The Basic Project Estimation Process

4.1 BASIC TOOLS FOR SOFTWARE ESTIMATION

a. Estimation Model Adopted

This work adopted the **CO**nstructive **CO**st **MO**del II (COCOMO II) developed by Boehm (1995), for the estimation of software projects.

We acknowledge the fact that, there are many software estimation models, such as Putnam Model, Top-down Model, Bottom-up Model, etc. Most of these models have some major weaknesses. For example, one significant problem of the Putnam model is that it expects the estimator to be able to estimate accurately the size (in lines of code) of the software to be developed. The uncertainty (especially in the early stage) in the software size can easily lead this model to produce inaccurate cost estimation. The Top-down model provides no detailed basis for justifying estimation results, and tends to overlook low-level software development components. The Bottom-up model on the other hand, has as its main weakness, the fact that, it may overlook many of the system-level costs (integration, configuration management, quality assurance, etc) associated with software development (Wu 1996). In the light of the foregoing, we adopted COCOMO II because of the following features, some of which are its major strengths over others:

- It focuses on issues such as non-sequential and rapid development process models (which are common in developing countries); reuse-driven approaches involving Commercial-Off-The-Shelf (COTS) packages, reengineering, applications composition, and application generation capabilities; object-oriented approaches supported by distributed middleware; software process maturity effects and process-driven quality estimation.
- It is parametric, and allows important aspects of the software project to be characterized by variables (or parameters).
- Once the values of these parameters are determined, the project can be estimated.
- It is ideal for modern technology and software process management.

b. Basic Model Equation

The COCOMO II basic model equation comes in two versions: Version 1 for Early Design Stage and Version 2 for Post-Architecture (Baik 1999).

$$PM = A(Size)^{1.01} + \sum_{j=1}^{5} SF_j \times \prod_{i=1}^{7} EM_i \qquad 1$$

$$PM = A(Size)^{1.01} + \sum_{j=1}^{5} SF_j \times \prod_{i=1}^{17} EM_i \qquad 2$$

where:
PM – Effort in Person-Months
A – Constant (kept at 2.45 as at now)
Size – Estimated size in KSLOC (Thousand Source Lines of Code)
SF – Scale Factor
EM – Effort Multiplier.

The early design model (Eq. 1) is used in the early stages of a software project, when very little may be known about the size of the product to be developed, the nature of the target platform, the nature of the personnel to be involved in the project or the detailed specifics of the process to be used. The early design model adjusts the effort using 7 EMs as stated in Eq. 1.

Equations 1 and 2 are similar, except for the 17 Effort Multipliers in Eq. 2. The larger number of EMs takes advantage of the greater knowledge available later in the development process. The post-architecture model (Eq. 2) covers the actual development and maintenance of a software product.

c. Determining Project Size

There are two basic approaches to determining project size; namely; the Lines Of Code (LOC) strategy and the Function Points (FP) strategy (Pressman 1992).
In this work we adopted the function points approach because of the following:

- Determining a line of code is difficult due to conceptual differences involved in accounting for executable statements and data declarations in different programming languages.
- LOC approach is very unrealistic during the early design stage, in which it is not yet known the language and platform of implementation, etc.
- The function point approach is based on the amount of functionality in a software project, and a set of individual project factors.
- It is a useful size estimator, since it is based on information available early in the project life-cycle.

d. Estimating Schedule

The schedule for a software development project can be obtained from the effort estimate. This generally involves estimating the number of people who will work on the project, what they will work on (the Work Breakdown Structure), when they will start working on the project and when they will finish (the staffing profile). Once this information is available, one needs to lay it out into a calendar schedule. Historical data from an organization's past projects or industry data models can be used to predict the number of people needed for a project of a given size and how work can be broken down into a schedule (Pressman 1992).

For most developing countries where software engineering practices are still at the infancy, historical data is seldom available. A schedule estimation rule of thumb (McConnell 1996) can be used to get a rough idea of the total calendar time required, as given in Eq.3 below:

$$Schedule = B(PM)^{1/B} \qquad 3$$

where

Schedule is in months.
PM is effort in person-months
B is a constant value adopted for the computation.

Opinions vary as to whether B should be 2.0 or 2.5 or 3.0 or even 4.0. Only by trying it out will an organization see which value works best for her. In this work (including the software tool), a value of 2.0 is used for B (McConnell 1996).

Because this research is aimed at supporting software project management in developing countries, like Nigeria, where historical data on software projects is hardly available, the schedule formula of Eq. 3 was adopted.

e. Estimating Cost

Many factors must be considered when estimating the total cost of a software project. In software cost estimation, emphasis is normally placed on labour (McConnell 1996; Jones 2002). The simplest labour cost – and the one used in this work – is obtained by multiplying the project's effort estimate (in hours) by a general labour rate (in Naira per hour).

However, a more accurate labour rate would result from using a specific labour rate for each staff position (e.g. Technical, QA, Project Management, Documentation, Support, etc.). Using this approach, the estimator would have to determine what percentage of total project effort should be allocated to each position. This can only work well if historical data is available (McConnell 1996).

f. Risk Management

Effective risk management transcends management's daily activities, such as activities that determine whether the project's budget remains intact or if some or all of it gets traded off to support higher priority or more politically sensitive projects. Continuous, proactive risk management can be performed by all participants on a project, from individual contributors to top management of the organization (ASC Corporation 2003).

Project risks are assessed for their risk exposure using Eq. 4 below:

$$RE = P(Ou) \times L(Ou) \qquad 4$$

where:
P(Ou) is the likelihood of occurrence;
L(Ou) is Potential Loss (impact);
RE is Risk Exposure.

As shown in Eq. 4 above, our work defines the Risk State using two quantitative values: (1) Probability and (2) Impact, and then the Risk Exposure is computed from the Probability and Impact.

5.0 THE SOFTWARE SYSTEM

The software system which is hereafter referred to as The Project Manager TPM V1.0 (code-named by the researchers) was designed to automate the software project management process. Its purpose is to provide a more effective way of managing software projects using online facilities offered by the software.

5.1 Architectural Style

As a prelude, it should be understood that, client and server as used in this work refer to software and not hardware entities. In its very fundamental form, the term client-server involves a software entity (client) making a specific request which is fulfilled by another entity (server). Figure 3 illustrates these client-server transactions.

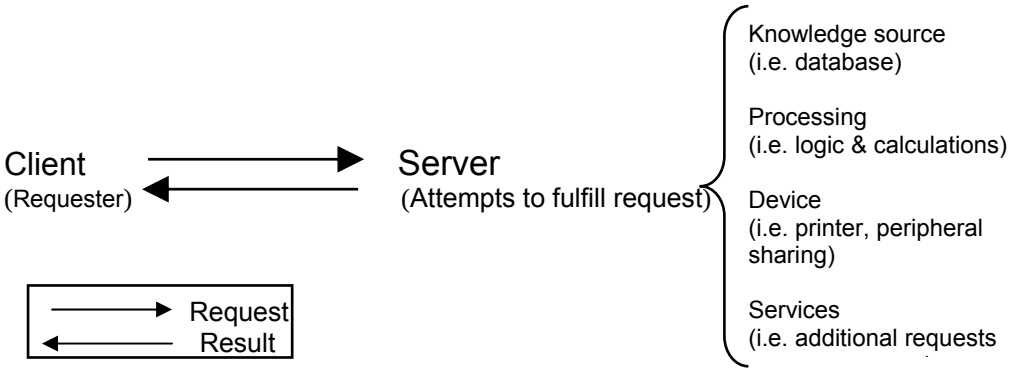

Figure 3: Client-Server Transactions

5.2 The Three-Tier Client-Server

The vast majority of end-user applications consist of three components: presentation, processing, and data. The client-server architecture is defined by how these components are split up among software entities and distributed on a network (Hunter and Crawford 2001).

The client handles only the presentation, and makes remote calls to the middle-tier functionality server, when calculations or data access is required. The middle-tier handles data access on behalf of the 1st- (presentation) tier to the third-tier (database). For this work, the middle-tier was coded in Java – a highly portable, flexible and non-proprietary language (Bayross 2001; Naughton 1999).

The researchers intend that, project team members would use their PCs (1st-tier) to make requests to the Java-based application server (2nd-tier), which in turn makes data access to a relational database management system (3rd-tier). The Presentation tier is designed in JavaScript (Wilton 2000; Flanagan 2002).

With this architecture, the structure of the database or even its implementation can change without affecting the 1st-tier (presentation). For instance, the database can be changed from Microsoft Access to Oracle without making any changes to the code that handles presentation (Siple 1998). Figure 4 shows a schematic representation of the three-tier system architecture developed in this work.

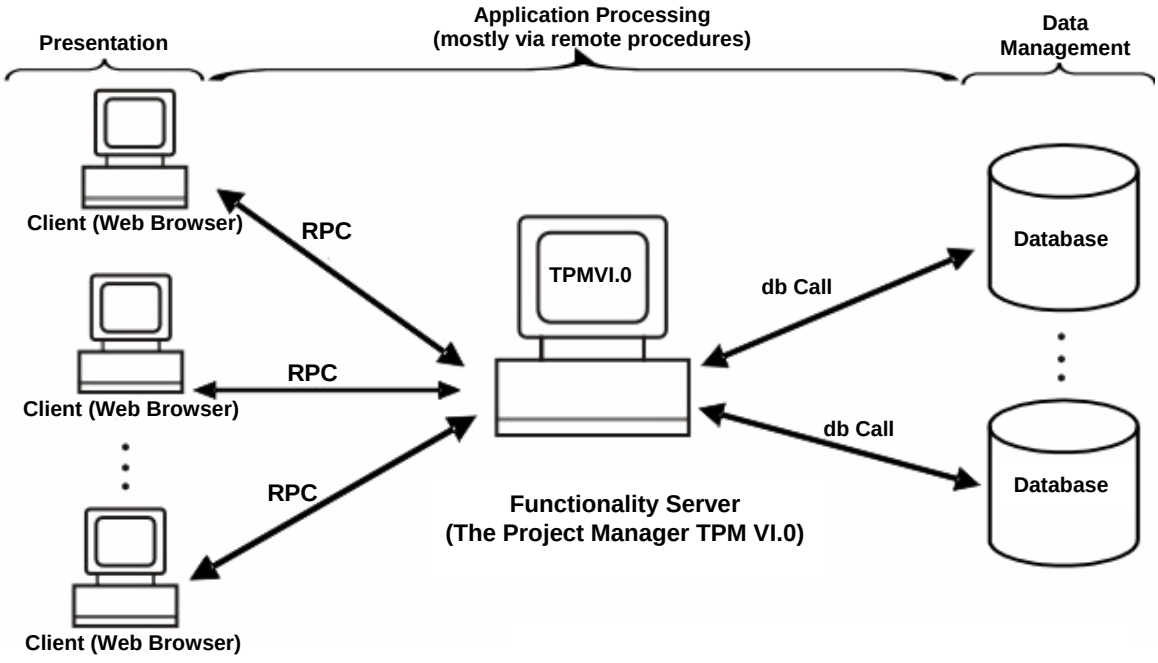

Figure 4: Three-tier Client –server architecture

In terms of the logical design for TPM V1.0, we adopted an object-oriented approach based on UML using class diagrams. Figure 5 is a Level 1 class diagram showing a detailed logical design of the system.

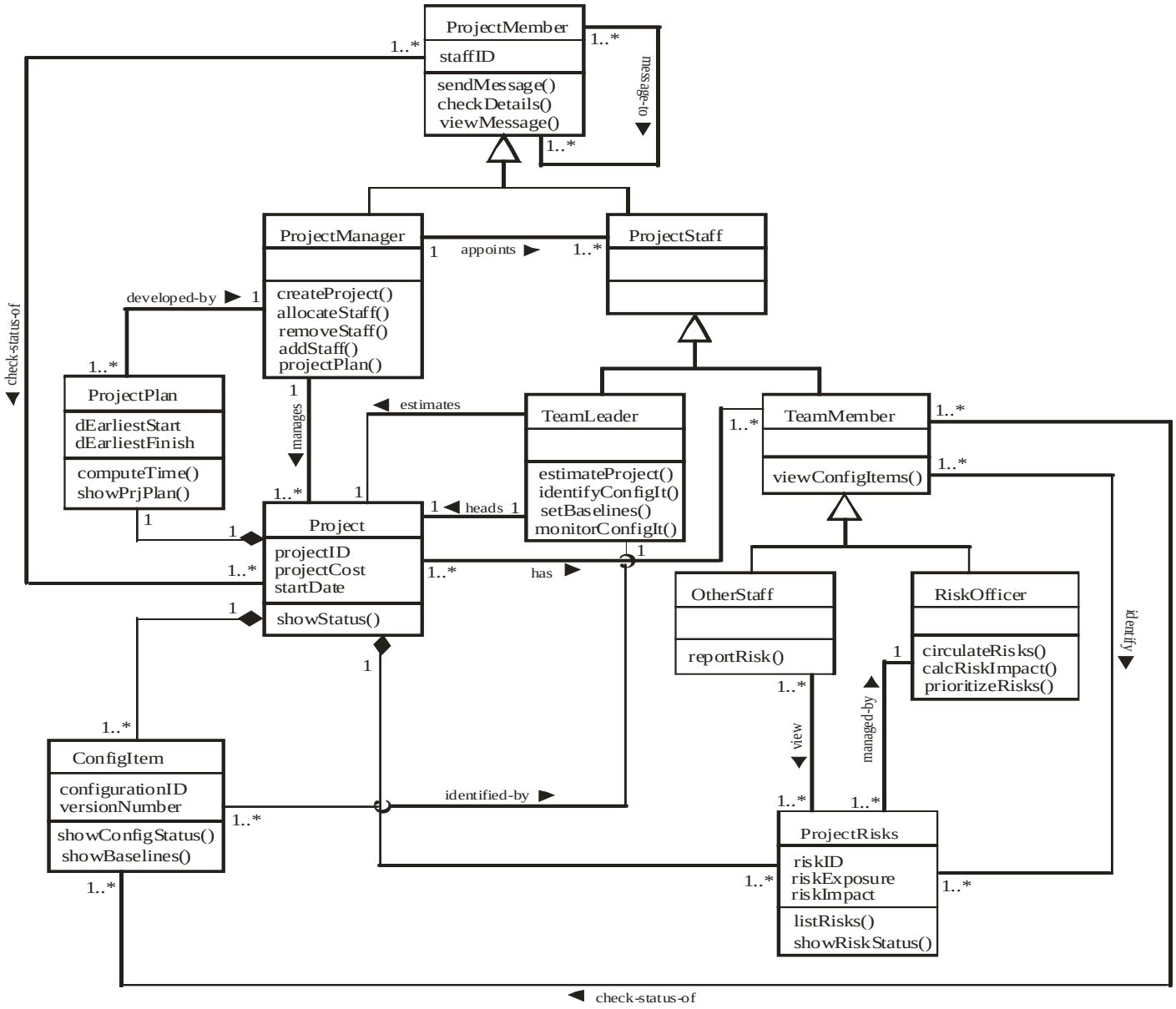

Figure 5: Level 1 Class Diagram of The Project Manager V 1.0

5.3 Target Server Requirements

TPM VI.0 is a web application which runs on a web server. For the application to run correctly, the target server machine must satisfy the following requirements.

- **The Web Server**
  The target server should have Apache Tomcat Web Server – Tomcat 3.2 and above installed. The Servlet API reference implementation of this version is written entirely in Java. It supports Servlet API 2.2 – the first Servlet API to support web applications. The development web server for TPM V1.0 is Tomcat 5.2.7.

- **Java Virtual Machine (JVM)**
  The target server machine should have a Java Runtime Environment that has a J2SDK or J2r1.4.1 or above version of Java Virtual Machine in order to be able to run servlets – the strength of web applications.

5.4 Implementation

As we have earlier stated, the software for this work was implemented using four sample projects. The figures and tables below show screen shots and some of the reports generated by TPM V1.0.

It should be noted that COCOMO II (the base model used in TPM V1.0) uses so many factors and elements in estimation. This is the reason that its accuracy is highly acceptable by many software practitioners (Baik 1999). As a result of these numerous estimation factors, project managers often find it too tasking and discouraging to go through the estimation process without forgetting (leaving out) some factors, which normally result in unacceptable delays in project estimation and planning. TPM V1.0 solves this problem by guiding the manager through the entire estimation and other project management processes, with minimal paperwork, and yet keeps all information about each project on the database. Access to project information is quick and easy. The input fields (Figures 6 and 7 below) show the range of data expected; and the corresponding data elements are prompted in ranges using dropdown lists.

Figures 6 and 7 shown as example of implementation are for the Early Design Stage. This enables the project to start. More realistic information is gathered during the Post-Architecture Stage and TPM V1.0 allows for the adjustment of the initial figures to obtain more accurate estimation. The interfaces for Early Design and Post-Architecture stages are similar.

TPM V1.0 is very dynamic. As inputs/outputs/enquiries and data elements change, the total function points, project size, effort and schedule are recalculated automatically. Figures 6 and 7 show the estimation process in the right order. Some of the other aspects of software project management, such as, planning and resource allocation, risk management, and adding new users to the project, are also shown in Figures 8 to 10. Some of the project management hardcopies generated from TPM V1.0 are shown on Tables 1 to 4 below.

In summary, the most important feature of TPM V1.0 is the simplicity with which software projects can be managed, and the accuracy of project estimation is reasonable because the underlying model (COCOMO II) is thorough. Because of its thoroughness, most practitioners prefer to use less-accurate, but simple models for project estimation, but with less reliability. TPM V1.0 encourages practitioners to do proper project management by leading them through the entire process using simple and meaningful interfaces. Less-experiences practitioners can also estimate and manage projects with it, hence improve their project management skills with the software.

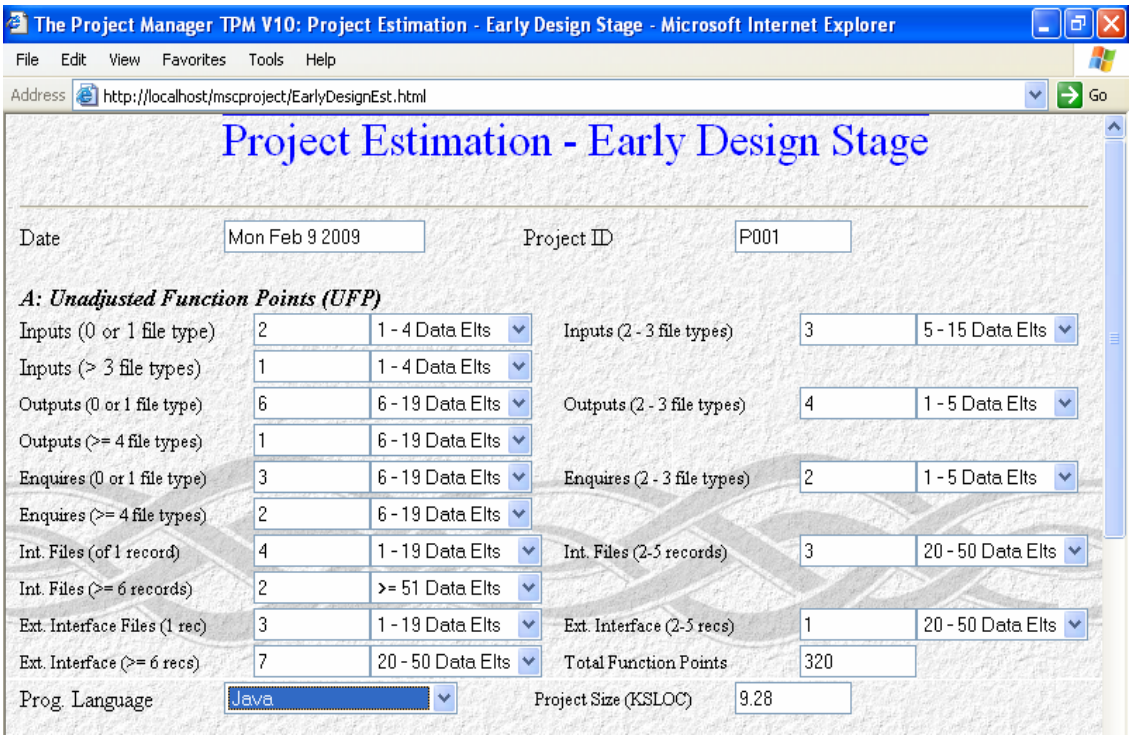

Figure 6: Project Estimation input for Unadjusted Function Points – Early Design Stage

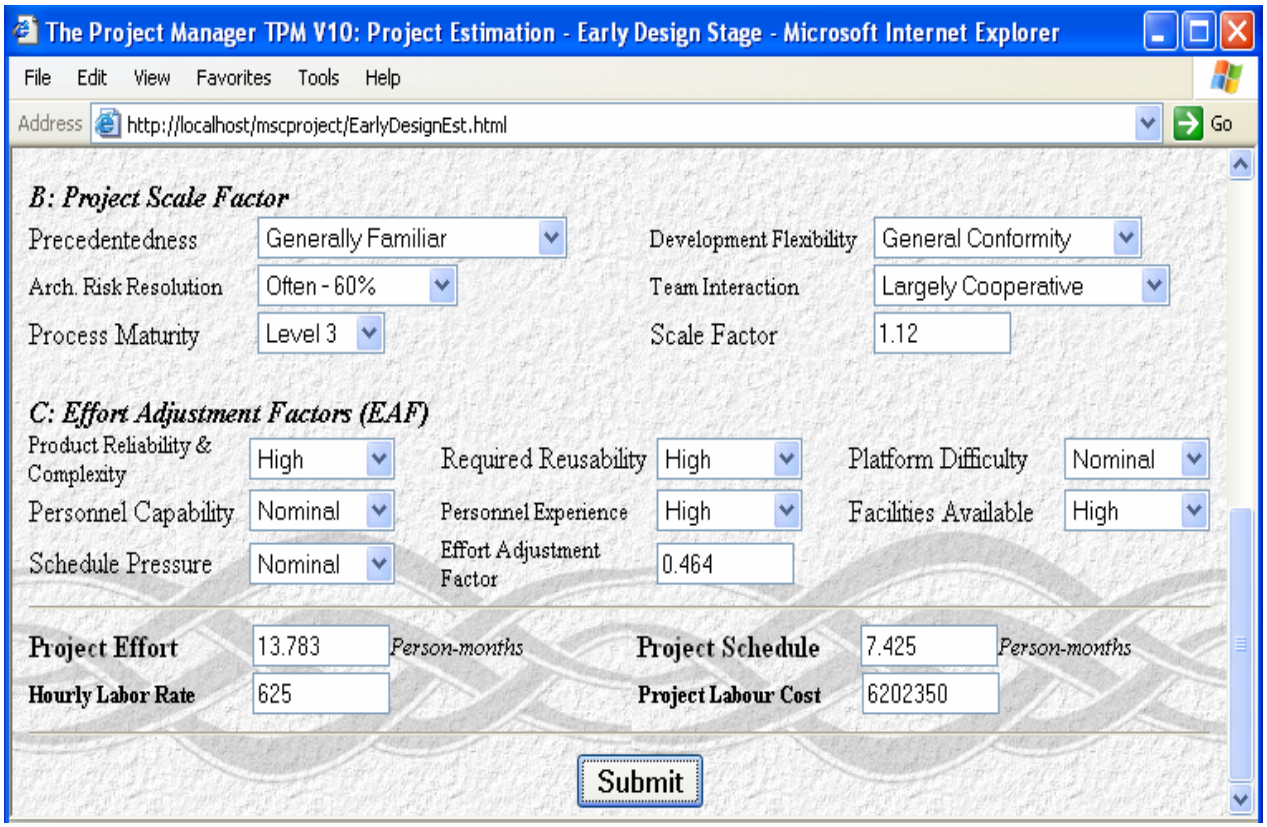

Figure 7: Project Estimation input for other sections – Early Design Stage

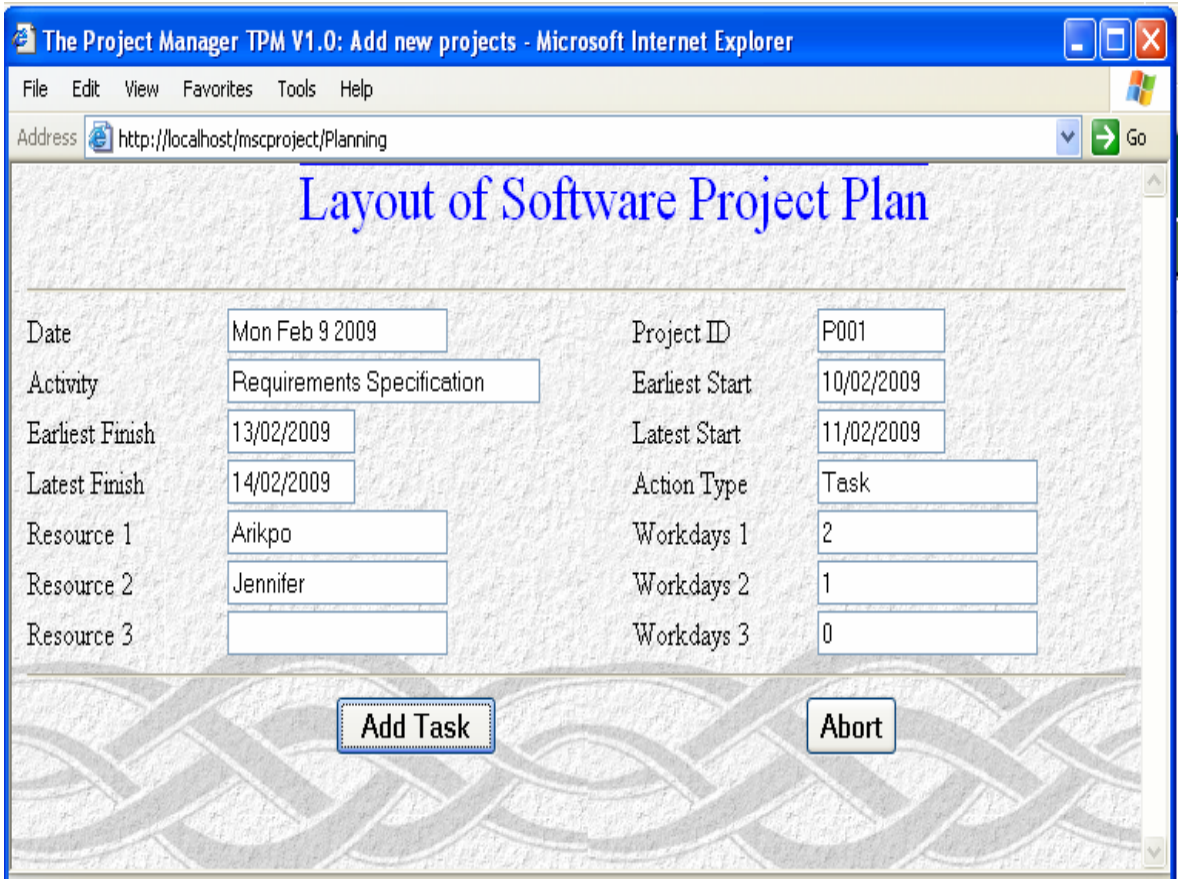

Figure 8: Software planning process showing resource allocation to activities

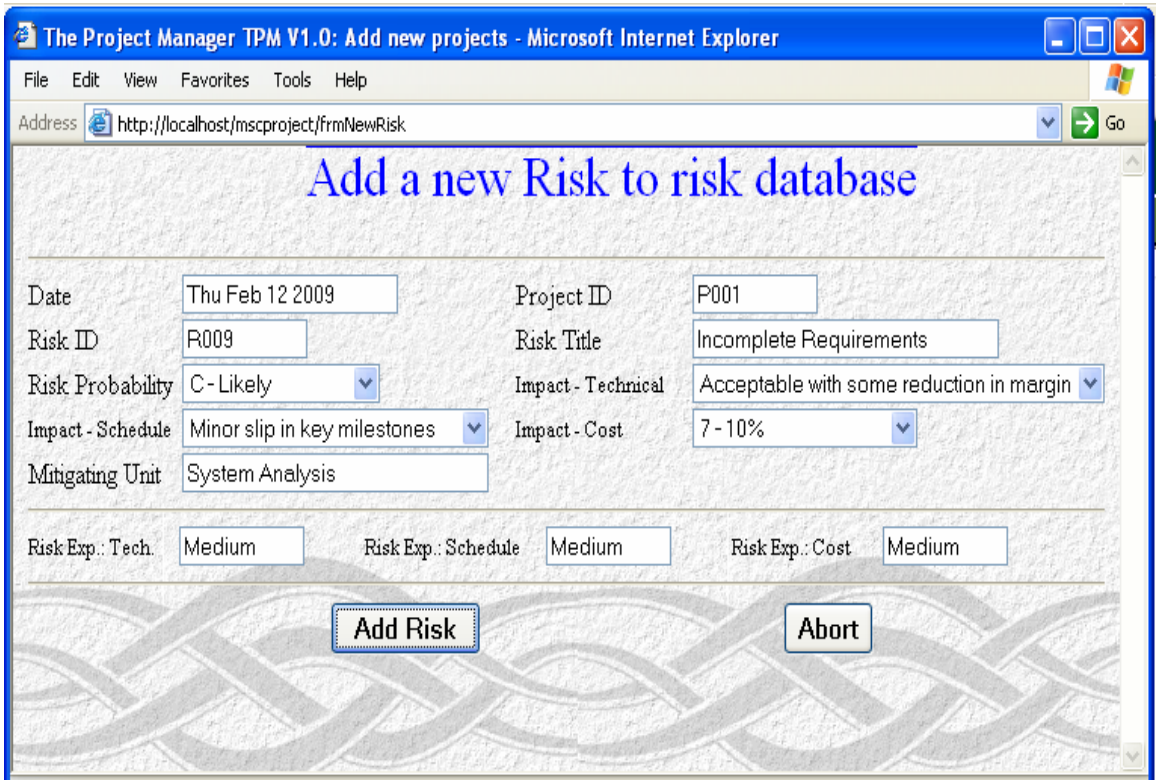

Figure 9: Software risk management (Exp. means Exposure)

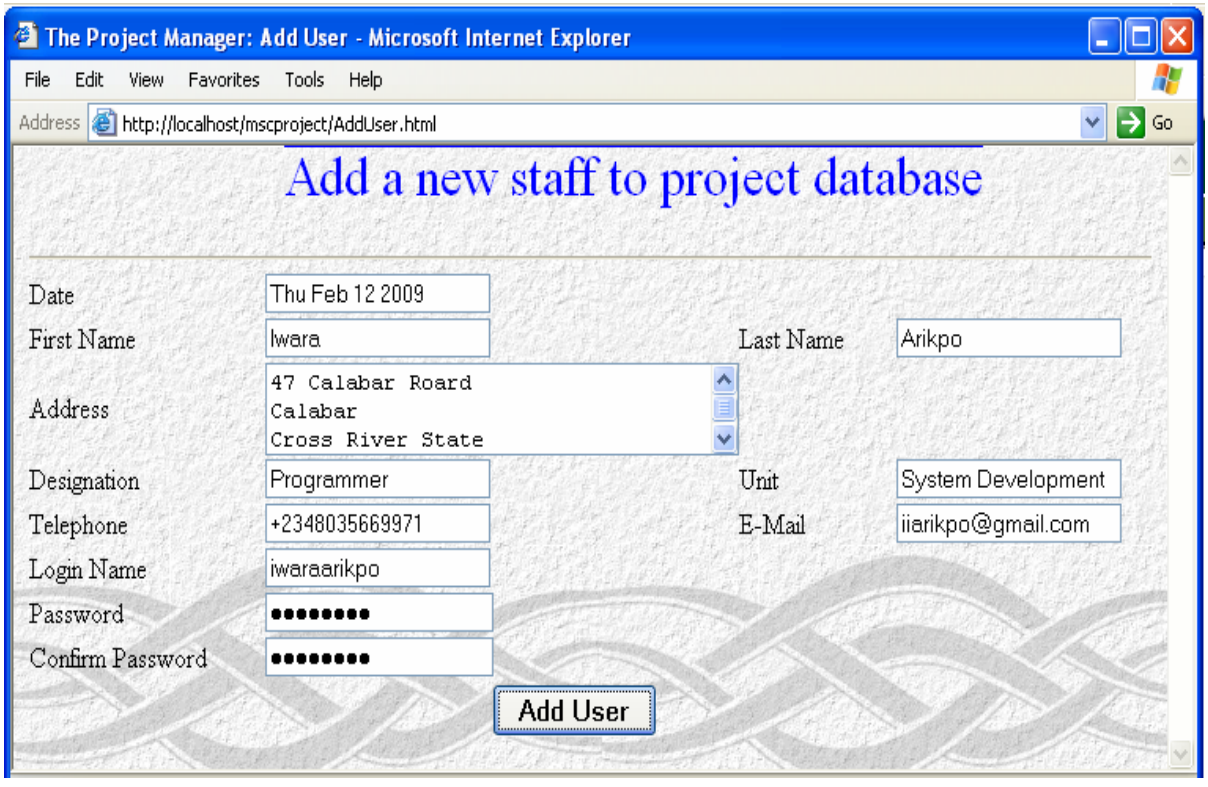

Figure 10: New user sign into the system

Table 1: Project Estimation output for Four (4) sample projects.

Project ID	Project Title	Estimation Date	Function Points	Project Size	Scale Factor	EffortAdj Factor	Project Effort (in P/M)	Project Schedule	Team Size	Hourly Labour Rate	Project Labour Cost (₦)
P001	Customs Revenue Collection	Fri May 13 2005	53	4.823	1.15	0.091	1.362	2.334	5	500	490,320.00
P002	Disease Analysis System	Fri May 13 2005	29	2.639	1.14	0.093	0.689	1.66	2	500	248,040.00
P003	Student Reg. & Result System	Fri May 13 2005	76	6.916	1.15	0.165	3.737	3.866	5	500	1,345,320.00
P004	Billing System Suite	Fri May 13 2005	37	3.367	1.13	0.316	3.052	3.494	4	400	878,976.00

Table 2: Project Plan report generated automatically by software

Activity	Latest Start	Latest Finish	Action Type	Resource 1	W_Days1	Resource 2	W_Days2	Resource 3	W_Days3
Initial customer contact	01/01/2005	02/01/2005	Starting task	Arikpo	2	Brian	2		0
Conduct FAST meeting	03/01/2005	05/01/2005	task	Matt	1	Brain	1		0
Review FAST results	08/01/2005	08/01/2005	task	Jennifer	1	Carolyn	1		0
Develop prelim product desc.	09/01/2005	12/01/2005	task	Arikpo	1	Matt	2		0
Perform feasibility econ. ana.	13/02/2005	16/02/2005	task	Mike	3		0		0
Perform system eng. task	15/01/2005	01/02/2005	task	Matt	6	Brain	6	Carolyn	6
Product description approved	15/01/2005	15/01/2005	milestone		0		0		0
Perform data design	25/05/2005	26/05/2005	task	Matt	4	Carolyn	4		0

Table 3: Output of Risk Management, generated automatically by software.

Risk ID	Risk Title	Mitigating Unit	Exposure Technical	Exposure Schedule	Exposure Cost
R001	Incomplete requirements	Analysis & Feasibility Study	Low	Medium	Low
R002	Delay in mobilization fees	Management	Low	Medium	Low
R003	Computer equipment breakdown	Technical	Medium	Low	Low
R004	Blackout in power supply	Technical	Medium	Medium	Medium
R005	Improper documentation	Documentation & Design	Nil	Medium	Medium
R006	Lack of software tools	Project manager	Low	Low	Low
R007	System breakdown	Management	Medium	Medium	Medium
R008	Lack of logistics	Management	Low	Low	Low

Table 4: Summary of projects as requested by team members

Project ID	Project Title	Team Size	Client Name	Client Address	Team Leader	ExpStartDate
P001	Customs Revenue Collection	5	General Bank Plc	Calabar	Mr. Iwara I. Arikpo	2004-12-04
P002	Disease Analysis System	2	Unical Medical Centre	Calabar	Felix Ogban	2003-05-06
P003	Student Reg. & Result System	5	Centre for Gen. Studies	Calabar	Samuel Oyong	2004-11-09
P004	Billing System Suite	4	Nigerian Ports Authority	Calabar	A. O. Ayodele	2004-04-08

6.0 DISCUSSION

The software project management tool – The Project Manager TPM V1.0 – developed in this research is largely applicable to large software organizations, for three of reasons:

i. Larger organizations are more likely to have enough "similar" projects to provide adequate historical data for any given project.
ii. A large organization may have large database of information that can aid consistent quantification, and minimize subjectivity.
iii. Human factors have been identified as very important to the successful management of software projects. Larger organizations may have formalized procedures for personnel and project evaluation which can provide quantified inputs when required**.**

Apart from the strengths of the underlying model (COCOMO II), and advantages of TPM V1.0 discussed in Section 6.4, the software has advantage over some of the most popular project management applications like Microsoft Project. Some of these are summarized in Table 5 below.

Table 5: Comparison of TPM V1.0 with Microsoft Project

S/N	The Project Manager TPM V1.0	Microsoft Project
1.	The software can work on any platform because of the underlying implementation language, Java.	The software is strictly Windows-based.
2.	It is purely software project-oriented	It is generic for general project management. The peculiarity of software is not taken care of.
3.	It is network based in design and implementation.	It is stand-alone.
4.	COCOMO II (the underlying model) is very thorough, covering all aspects of a software management process.	Because of the generic nature, the underlying model does not cover all aspects of software management process.
5.	It provides such facilities as can help project-wide communication. For instance, a team member can place a message for a meeting on a given project, and others can access it.	N/A

CONCLUSION

This work developed a software tool, TPM V1.0, for automating the management of software projects in a country like Nigeria, where the software industry is very young, in terms of development.

SUGGESTIONS FOR FURTHER RESEARCH

This research has focused on the management of application software, and within a narrow domain – Nigeria. The other directions, in which further research can be extended, are the management of system and real-time software projects, respectively, in developing countries. Both of these directions require the identification and involvement of more abstract features to support a more broad-based project management.

In the area of estimation, an accurate measure of project size is not available early in the development life cycle. So it would be more pragmatic if further research could be shifted from early cost estimation to the estimation of a productivity coefficient that describes the effect that certain cost drivers and environmental factors – such as power supply, resource availability, etc. – have on productivity.

REFERENCES

ASC Corporation., 2003. Risk Management Process and Implementation – Practice Book Number One: Overview and Guidance. American Systems Corporation, 13990 Parkeast Circle Chantilly, USA.

Baik, J., 1999. USC COCOMO II 1998.0 Model Definition Manual. University of Southern California Center for Software Engineering, California.

Bayross, I., 2001. Web Enabled Commercial Applications Development Using... Java 2. Tech Publications PTE Ltd., Singapore.

Boehm, B. W., 1995. Constructive Cost Model II. University of Southern California Center for Software Engineering, California.

Feigenbaum, E.A., and McCorduck, P., 1983. The Fifth Generation. Addison-Wesley Longman, Inc. USA.

Flanagan, D., 2002. Java Script – The definitive Guide. 4th Ed. O'Reilly and Associates, Inc., USA.

Gibbs, W. W., 1994. Staff Writer. Scientific American, USA.

Hunter, J. and Crawford, W., 2001. Java Servlet Programming. 2nd Ed. O'Reilly & Associates Inc., USA.

Jalote, P., 2000. It's quality that has done the trick – Developing Nations in Asia & Latin America Want to Replicate India's story. Computer Science & Engineering 11T Kanpur, India.

Jones, C., 2002. Software Cost Estimation in 2002. Software Productivity Research Inc. Artemis Management System, USA.

McConnell, S., 1996. Rapid Development - Taming Wild Software Schedules. Microsoft Press Redmond, WA, USA.

Moder, J. J., Phillips, C. R. and Davis, E. W., 1983. Project management with CPM, PERT and PRECEDENCE Diagramming. 3rd Ed. Van Nos Reinhold, Atlanta, GA.

Naisbitt, J., 1982. Megatrends. Warner Books, New York.

Naughton, P. and Schildt, H., 1999. 3rd Ed. Java 2–The Complete Reference. Osborne/McGraw-Hill, California.

Page-Jones, M., 1985. Practical Project Management. Dorset House. USA.

Pressman, R. S., 1992. Software Engineering: A Practitioners Approach. 3rd Ed. McGraw-Hill, Inc., USA.

Radhakrishnan, N., 2004. Building Successful Software Companies in Developing Countries: The Case of India and Infosys. Infosys Technologies Limited, India.

Royce, W., 1998. Software Project Management: A Unified Framework Addison Wesley, USA.

Schach, S. R., 1996. Classical And Object-Oriented Software Engineering. 3rd Ed. Irwin/McGraw-Hill, USA.

Siple, M. D., 1998. The Complete Guide to Java Database Programming – JDBC, ODBC & SQL. The McGraw-Hill Companies Inc., USA.

Tessler, S. and Barr, A., 1997. Software R&D Strategies of Developing Countries. Stanford Computer Industry Project, UK.

Tomayko, J. E., 1989. Software Project Management, SEI Curriculum Module SET –CM -21 – 1.0. Carnegie Mellon University, USA.

Wilton, P., 2000. Beginning JavaScript. Wrox Press Ltd., Birmingham, UK.

Wu, L., 1996. The Comparison of the Software Cost Estimating Methods. University of Calgary, Canada.