International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:8, No:9, 2014

Inadequate Requirements Engineering Process: A Key Factor for Poor Software Development in Developing Nations: A Case Study

K. Adu Michael, K. Alese Boniface

***Abstract*—**Developing a reliable and sustainable software products is today a big challenge among up–coming software developers in Nigeria. The inability to develop a comprehensive problem statement needed to execute proper requirements engineering process is missing. The need to describe the '*what*' of a system in one document, written in a natural language is a major step in the overall process of Software Engineering. Requirements Engineering is a process use to discover, analyze and validate system requirements. This process is needed in reducing software errors at the early stage of the development of software. The importance of each of the steps in Requirements Engineering is clearly explained in the context of using detailed problem statement from client/customer to get an overview of an existing system along with expectations from the new system. This paper elicits inadequate Requirements Engineering principle as the major cause of poor software development in developing nations using a case study of final year computer science students of a tertiary-education institution in Nigeria.

***Keywords*—**Client/Customer, Problem Statement, Requirements Engineering, Software Developers.

I. Introduction

REQUIREMENTS Engineering is an all encompassing process of producing a comprehensive document in a natural language (not any programming language) that contains a description of only what a system is expected to do. The end product of sound requirements engineering is Software Requirement Specification (SRS). This is expected to be the working document for the design and implementation of the system. However, Problem Statement is the ingredient for the process of requirements engineering to produce the needed guiding document for a successful software product. Requirements engineering involves a systematic investigation into an existing system or process, and studying of materials and sources in order to establish facts and reach new conclusions. It is normally borne out of a problem, and hence the input to it is the problem statement. A 'Problem Statement' is a description of a difficulty in a system or lack of needed resources that need to be solved or at least researched into, and see whether a solution can be found. It can also describe the gap between the real and the desired or contradiction between principle and practice. According to [2], the importance of complete, consistent and well documented software requirements is difficult to overstate. The process is very important for up-coming software developers to understand in great detail. There is no greater risk to a software project than incomplete, misunderstood, or under-emphasized software requirements. Software requirements and the efforts undertaken to gather them are very germane to successfully understanding the users' problems and addressing these problems in a way satisfactory to them. Without requirements, there is no frame of reference and no benchmark to measure success. The process is illustrated in Fig. 1.

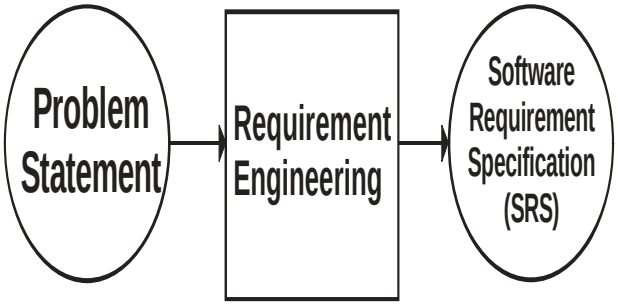

Fig. 1 Software Requirements Specifications (SRS)

The ultimate goal of writing a problem statement is to transform a generalized problem that bothers the users into a targeted, well-defined statement that can be resolved through a process of requirements engineering. It clearly identifies the purpose of the project. In most developing nations especially Nigeria, up-coming software developers have not been able to capture the local markets in terms of quality, basic and needed software product deliverables. One of the most important aspects of software design is the quality of the finished product. The finished product quality is measured by the users and it is measured by how well it satisfies their needs. The users' needs will be exposed via requirements and input that they provide to the development team which is the problem statement. These requirements and the problem statement should be provided throughout the design and development of the software system. Without the path provided by this information, the development team will quickly run off track and provide a product that does not satisfy the users' problems and needs. As [7] stated, a critical consideration of

M. K. Adu is a Senior Lecturer with the Computer Science Department, the Federal Polytechnic, Ado Ekiti, Nigeria (phone: +2348066714060; e-mail: memokadu@yahoo.co.uk).

B. K. Alese is an Associate Professor with the Computer Science Department and Acting Dean, Students' Affairs, the Federal university of Technology, Akure, Nigeria (phone: +2348034540465; e-mail: bkalese@futa.edu.ng).

information systems is that they satisfy users' needs. The experiences of students who are up-coming software developers in the field during their final year projects and reports gathered from some of the case study organizations are presented in Tables I and II respectively. This is to harmonize their views in order to bring out the exact cause of the inadequate requirements engineering process and consequently be able to propose holistic solution to the problem. Few of the organizations affirmed that home grown software products were usable, but they believed that they were not so reliable and maintainable. They opined that they were delivered without necessary documentations. They also established the fact that up-coming software developers were in a hurry to code rather than taking enough time to interact with the users and interpret their intentions as a problem statement.

TABLE I
SUMMARY OF STUDENTS' ACTIVITIES/EXPERIENCES DURING SOFTWARE DEVELOPMENT PROCESS WITH SOME CASE STUDY ORGANISATIONS

Reg No	Project topic	Case study	No of visit	successful interview conducted	No. of questionnaires administered	Responses	Able to get enough facts for software development?
CS/2-206	Development of Inventory System.	Memolinks Computer System, Akure, Nigeria	7	2	50	15	NO
CS/2-017	Development of Course Allocation System	Federal Polytechnic Ado-Ekiti, CEC.	10	1	-	-	NO
CS/2-062	Development of Drug Administration System	Eben Pharmaceutical, Akure, Nigeria.	6	2	-	-	NO

TABLE II
SOME ORGANISATION'S ASSESSMENTS OF DEVELOPED SOFTWARE PRODUCTS BY UP-COMING DEVELOPERS

Assessment Factors	Memolinks Associates	Eben Pharmaceutical Company Limited	CEC, Federal Polytechnic, Ado-Ekiti, Nigeria.
Usability	Fail	Pass	Pass
Reliability	Fail	Pass	Fail
Maintainability	Pass	Fail	Fail
Availability of Relevant documentations	Fail	Fail	Fail

II. INDENTIFIED CAUSES OF INADEQUATE ENGINEERING PROCESS

There is no real step-by-step guide that leads to failure of software development project among these up-coming developers; however, there are many factors that contribute. They include and are not limited to the following:

A. Lack of End User Involvement

Users can be very busy people with regular day jobs. They are not always given sufficient time to devote to the software development team. Without involvement from the end users (or their representatives), requirements specification and product quality are likely to suffer and the user community may feel less committed to the end product. If the users are not sufficiently committed or if they cannot clearly articulate what they want, then there will be limited ways to assure the quality of deliverables.

B. Poor Planning and Estimation Processes

A project that is poorly planned and estimated is risky, difficult, and can lead to system failure. Without a decent plan and estimate, resources cannot be managed and organized, risks cannot be mitigated, dates and budgets cannot be forecasted, effective reporting cannot take place, and the measures of success will be flawed from the outset.

C. Failure to Effectively Manage Changes to Scope

Incorporating necessary scope changes into a system is often a prerequisite for delivering a fit-for-purpose product. If scope is not controlled, changes will creep in unnoticed and quality may be adversely affected.

D. Inadequate Man Power Resources

A system can fall apart quickly if the developers are not sufficiently skilled.

E. Too Long or Unrealistic Time Scales

When system development becomes too long, the project can lose momentum or one can end up delivering products and services that are no longer of any benefit to the customer and organization. On the other hand, senior managers may set unrealistically short project time scales in an attempt to speed up delivery without considering the volume of work that needs to be done. These anomalies can either lead to the system being delivered late, or a significant amount of features being cut out.

F. Failure to Adequately Identify and Document Requirements

Some software development processes have high-level, vague, or poorly documented requirements. If design is kicked off too early without the core requirements having been adequately identified, documented, and agreed on by the users, the software so developed has every chance of producing the wrong product and consuming more time and money during testing and rework phases than planned.

III. SOLUTION THROUGH ADEQUATE REQUIREMENTS ENGINEERING PROCESS

Requirements are the "what" of a system. A requirement is a feature of the system (being designed) or a description of something the system is capable of doing in order to fulfill the system's purpose [6]. It is further defined as a basic need of a

system to perform or facilitate a user's particular work or task. Both natural need and diligent effort cause software requirements to surface in a software project. Different types of requirements exist as well. A functional requirement is one that is specific to a particular software need and that can be addressed directly via code. Functional requirements define acceptable states that a system can be in [6]. Non-functional requirements define issues of performance, security, and hardware considerations.

Requirements Engineering is a process of producing one comprehensive document in a natural language and contains a description of what the system will do. It does not specify how it will be done. The input to requirements engineering is the problem statement prepared by the user of the system. If there is already a computer based system in existence, then problem statement will give an overview of the existing system along with expectations from the new system. Therefore, for an effective system to be developed, the software developer must religiously follow the prescribed steps in requirements engineering as illustrated in Fig. 2.

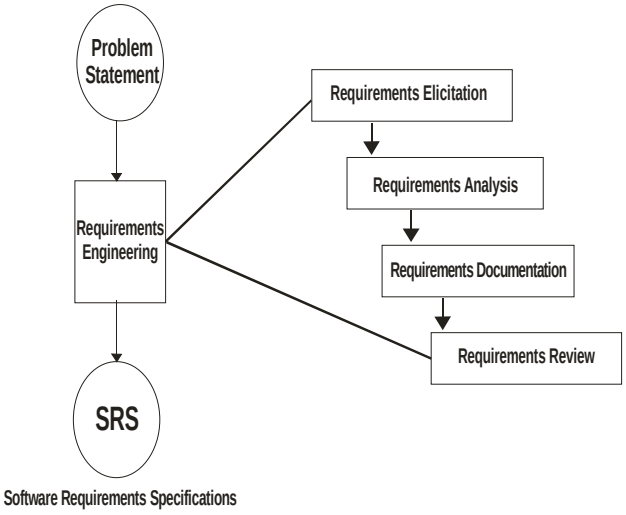

Fig. 2 Steps of Requirements Engineering

A. Requirements Elicitation

This is the process of gathering the requirements from the users or the customers in order to have first hand information on the proposed system [1]. It is always good that the developer does not assume anything but get the details from the end users. Every program solves a problem. A tax return program solves the problem of organizing and filing taxes. A word processor solves the problem of writing, editing, formatting, and printing text. Even a video game solves the problem of keeping people amused. A program is only as useful as the problem it solves. Most programs simplify and automate an existing problem, such as a money management program that simplifies organizing and paying bills instead of using paper and an adding machine [5]. The goal of any program is to make a specific task faster, easier, and more convenient. The only way to reach that goal is to identify what task your program is trying to solve in the first place. The summary of all these is that it is about identifying the problems.

B. Requirement Analysis

The requirements are analyzed in order to identify inconsistencies, defects and omissions [4]. If a software product is for personal use, one can make a program look and act any way he wants, just as long as he knows how to make it work. But if such application is planned to be given or sold to others, there is need to know who is going to use it. Knowing your program's typical user is critical. If users do not like the program for any reason, they are unlikely to use it. Whether the program works or not is often irrelevant. By designing your program with the user in mind, then there is hope that customer will buy a copy. Even if you write a program that works perfectly, users may ignore it because they don't like the way it looks, they don't understand how to give it commands, it doesn't work the same way as the old program they currently use, the colours do not look right to them, and so on. The goal is to make your program meet your users' needs, no matter how bizarre or illogical they may seem. After you identify the user, you need to know what type of computer the user intends to run the program on. The type of computer that your program runs on can determine which computer languages you can use, the hardware that your program can expect to find, and even the maximum size of your program [3]. When designing any program, consider your programming skill as well.

C. Requirements Documentation

This is the end product of requirements elicitation and analysis. The documentation is very important as it will be the foundation for the design of the software. The documentation is known as Software Requirements Specification (SRS).

D. Requirements Review

The review process is carried out to improve the quality of SRS. It may also be called as requirements verification.

IV. Software Re-Engineering

It is a common practice among up-coming software developers to simply adapt a software system of an organization *A* for another organization *B* simply because of perceived similarities in operations. As an example, in educational institutions where a developer has successfully developed a software package for an operation, he simply changes the headings and adopts same for another institution because same operation is involved, say Students' Results Processing. This is wrong. There is need for adequate re-engineering to suite the specific and peculiar needs of the new organization, even if there is need for any adaptation of some similar feature. Software Re-engineering is concerned with taking existing legacy system and re-implementing them to make them suitable and maintainable either for present or new users. This is illustrated in Fig. 3.

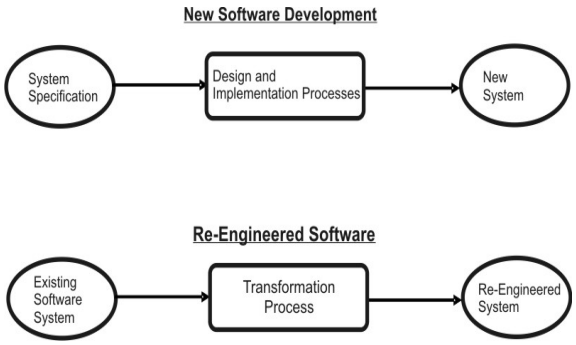

Fig. 3 The Difference between the processes of new software development and software Re-engineering

V. Software Design

The next stage which is Design phase is made simple through adequate requirements engineering process. Design is a highly significant phase in the software development process where the designer plans "how" a software system should be produced in order to make it functional, reliable and reasonably easy to understand, modify and maintain. A Software Requirements Specifications (SRS) documents tells us '*what*' a system does, and becomes input to the design process, which tells us '*how*' the software system works. Designing software system means determining how requirements are realized and the result is the Software Design Document (SDD). Therefore, the purpose of design phase is to produce a solution to a problem given in SRS document.

Conceptual and Technical Designs

The process of software design involves the transformation of ideas into detailed implementation description, with the goal of satisfying the software requirements. The designer must satisfy both users and the system builders (the programmer). The user understands what the system is to do, while the system builder on the other hand understands how the system is to work [1]. For this reasons, the design must be a two part iterative process. The first is the Conceptual design that tells the customer/user exactly what the system will do and the Technical design that allows system builder (the programmer) to understand the actual hardware and software needed and the logical steps to solve the customer/user problem. This is illustrated in Fig. 4.

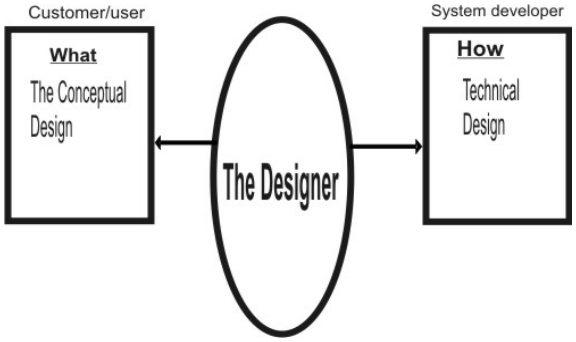

Fig. 4 Dual-process Software Design

VI. Conclusion

It is alarming that significant numbers of software development projects by up-coming programmers still fail to deliver benefits on time and to expectations in developing nations like Nigeria, in a world that is being driven by information technology of which software is a major component. Many have attributed this to quality of teaching process in information technology. However, beyond this is inability of the software developers to fully integrate the process of requirements engineering in their efforts at developing a reliable software products. Students should be good enough to develop a sellable software package after graduation in higher educational institutions. This can only be achieved by assisting them to get the needed inputs for their research works. Also it should be impressed on the various organizations that the young software developers are out to solve problems that could add value to the organizational operations of the users/customers. Towards this objective, this paper has discussed the issues that many studies confirm as significant causes of failure in software development projects by attempting to draw inference from some selected students' projects from the department of Computer Science, Federal Polytechnic, Ado-Ekiti, Ekiti State, Nigeria. The overall process of requirements engineering is explained and the importance is clearly elucidated for clear understanding of its application for developing reliable and marketable software products. The paper has brought to fore the major factors responsible for poor software development among up-coming software developers in developing nations as incomplete requirements specifications as a result of poor involvement of the users.

Acknowledgment

Our thanks go to the department of computer science, the Federal Polytechnic, Ado-Ekiti, Nigeria, the students of the department and the case-study organizations of the students' projects for their co-operations.

References

[1] K.K. Aggrwal, S. Yogesh, Software Engineering. 3rd ed. New Delhi: New Age International Publishers Limited, 2008.
[2] M. Early, Relating software requirements to software design. SIGSOFT Software Engineering Notes, 11(3), 37-39. 1986
[3] J.A. Hoffer, J. F. George, & J.S. Valacich, Modern systems analysis and design, 2005.
[4] J. Karlsson & K. Ryan, Supporting the Selection of Software Requirements. Proceedings of the 8th International Workshop on Software Specification and Design, 146, 1996
[5] D. Martin, J. Rooksby & M. Rouncefield, 'Users as contextual features of software product development and testing'. Proceedings of the 2007 international ACM conference on Conference on supporting group work, 301-310, 2007.
[6] S.L. Pfleeger, Software Engineering. Theory and Practice. Prentice Hall, 2001.
[7] J. Verner, K. Cox & S.J. Bleistein, 'Predicting Good Requirements for in-house Development Projects.' Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, 154-163, 2006.