Improving Software Quality through Requirements Elicitation

Sereen Abu Aisheh
Faculty of Technology and Applied Sciences/ Al-Quds Open University, Nablus, Palestine
sabueshih@qou.edu

Abstract
Today's IT challenge is to deliver, as quickly as possible - and within a fixed budget, quality, business-critical software systems that can support business initiatives in a changing business environment, this means that three factors should be maintained to produce the desired software, these would be cost, quality and time.

Most project management efforts concentrate on meeting time and cost constraints, passing over the quality factor, research in software development industry shows that the major problem facing software development isn't crossing time and budget limits (though it's a big issue), but it's the production of software systems that wont be used as they don't address vital business needs, this is definitely a quality issue.

This paper focuses on improving the quality of a software products through improving the requirements elicitation process, it employs the results of a research conducted in two local software houses to reveal the relationship between software quality and requirements elicitation process as the road to producing better software, it also discusses two issues that may lead to failure in elicitation process (bad communication and requirements volatility).

Keywords: Requirements Elicitation, Evolving Requirements, Software Quality.

Introduction

IT industry still has a gap between what clients need and what they really get, most software development projects are never completed because they run out of budget and time; and even completed ones are of poor quality.

According to Chaos Report 2003 for example, clients only got 54% of the functions they requested, and 42% of the delivered functions were unused for long periods, the cause of these shortcomings was attributable to changing user requirements.
As the report indicates, there is a big gap between theory and practice in requirements area, software developers have many tools and procedures for managing client requirements and translating them into a working software, but these tools are rarely used; as the time-to-market pressure increases, most companies tend to put less efforts in the area of requirements management, which will in turn affect the overall product quality. Statistics show that poor quality has negative effects on the long run specially for budget; the less the product quality is, the more modifications it needs, modifying the product means more time and cost, which may make the overall development project unprofitable for both client and developer.

Most developers seek the solution for this defect in the area of production tools and procedures; they focus on using some brand technologies, while the real problem lays in the requirements management process, like all other projects, software development projects need good project management process to produce a 'quality' product that meets the client's demands without crossing time and budget constraints, this means that there are predefined deliverables that should be produced after each development activity, but what if those deliverables are not what clients really want? What if they couldn't be produced?

In software development, deliverables represent the desired system functionalities that were specified based on the elicited client requirements, so if the wrong requirements were elicited; the whole project will be a failure as a result and the final product may not be used at all. To prevent such failures, more efforts should be put in order to enhance the requirements management process.

According to SQS report 2006; the problems in software development projects that failed or needed considerable additional efforts to be completed were [13]:

- Shortcomings in the specification of the requirements (50%)
- Shortcomings in the management of client requirements (40%)

The reason of failure is because most companies don't have a structured procedure for requirements management, or some do have standardized procedures but those are not implemented consistently in practice. In a research conducted for the purpose of this paper with 100 IT employees in two major IT companies in Palestine, 62% of the interviewed IT personnel stated that their companies don't have well-defined procedures to understand clients' needs, 33% of them stated that their companies do have such procedures but those are rarely implemented in software development projects.

In addition, SQS statistics show that clients are paying additional 20% of the original contract value in average for changing requirements, our research results supports this finding as 40% of interviewed IT personnel admitted that in most cases their client asked for changes after the system was delivered, which entail more time and cost, and affected the overall quality of the product in turn.

Requirements Elicitation

Requirements elicitation is the first stage in building an understanding of the problem that the software is to solve [10]. Technically, elicitation is a process where clients, users, and developers reveal and articulate their requirements [14] but it doesn't mean that requirements are all there and can be easily captured by using any appropriate technique [11]. Most requirements management methods presume that requirements are explicitly and completely stated; however, experience shows that requirements are rarely complete and usually contain implicit requirements, software requirements characteristically suffer from inconsistency, incompleteness, ambiguity, duplication, and inconstancy [12], the way to overcome the fuzziness of requirements is by applying a structured elicitation process that deals with fact-finding, information gathering, and integration in order to obtain a set of requirements which describe the characteristics of the possible solution(s) [1].

Requirements Elicitation Problems

Problems of requirements elicitation can be grouped into three main categories [1, 12, and 14]:

- Problems of scope, in which the requirements may address too little or too much information (i.e. defining the boundary of the system).

- Problems of communication between the communities participating in the development process (e.g. users, stakeholders, and developers):

- Problems of volatility: the changing nature of requirements as they evolve over time which represents the main obstacle in elicitation process.

Our interest here is in studying the effect communication and volatility problems on the elicitation process and hence on software quality, next we discuss these problems in more details.

Problems of Volatility

Requirements change [6, 11. 14, 2]. During the time it takes to develop a system users' needs may mature because of increased knowledge brought on by the development activities, or they may shift to a new set of needs because of organizational or environmental pressures. If such changes are not accommodated, the original requirements set will become incomplete, inconsistent with the new situation, and potentially unusable because they capture information that has since become out of date.

One primary cause of requirements volatility is that user needs evolve over time. The requirements engineering process of elicit, specify, and validate should not be executed only once during system development, but rather should be returned to so that the requirements can reflect the new knowledge gained during specification, validation, and subsequent activities. Requirements management process should be iterative in nature, "so that solutions can be revised in the light of increased knowledge" [1].

Another cause of requirements volatility is that requirements are the product of the contributions of many individuals that often have conflicting needs and goals. Due to political climate and other factors, some times the needs of a particular group may be overemphasized in the elicitation of requirements. Later prioritization of the elicitation communities' needs may correct this mistake and result in requirements changes. Both the traceability of requirements and their consistency may be affected if these changes are frequent and not anticipated [1, 4].

Organizational complexity is another cause of requirements volatility as organizational goals, policies, structures, and work roles of intended end users all may change during the system's development, especially as the number of users affected by a system's development increases.

Problem of Communication

Requirements management is a social process [1, 9] it involves various communities with different backgrounds and needs, any elicitation process that ignores this social factor will absolutely fail in understanding the characteristics of the future software.

One factor that may influence the degree of understanding is language; if clients and developers speak different languages, then the probabilities of misunderstanding what clients really want are maximized. Another factor that disrupts effective

communication is the way clients express their demands, since they don’t have much knowledge in computer domain so they can't articulate their needs in a form that can be understood by developers.

Problems of communication and requirements volatility proved to be critical issues as they may lead to building unsatisfactory software in the long run, our research reveals that difficulties in communication with client negatively affected the development process; 32% of respondents think that when the requirements management sessions (like JAD) were ill-structured, they had troubles understanding what their client really want, 17% of those also think that their companies didn’t spend enough time and efforts in the requirements definition activity as they met their client few times only at project start. When asked about language difference between client and developer, 30% stated that when they were involved in projects for foreign clients, language difference was an obstacle as they couldn’t understand client requirements and in some cases those requirements were interpreted incorrectly. 52% of respondents also stated that most clients can't speak for them selves or they don’t really know what they need which in turn caused some requirements to be missing.

Respondents also said that in almost every project, clients keep changing their minds, they ask for a lot of modifications especially for functional requirements, 86% of respondents stated that the requirements statement was updated frequently due to changing client requirements, which caused the document to be inconsistent, 74% stated that the frequent modifications made it harder for them to design the target system as they were 'lost' and ultimately their client was unsatisfied with the final product.

Obviously, research result indicate that bad communication and evolving requirements can cause the requirements management process to fail, since the actual requirements which are the base of development process can't be elicited and documented, or the wrong requirements were defined and built, in both situations the resulted software was of poor quality as 38% of respondents declared, because its either not what clients expect and need, or it didn’t provide all demanded functions.

Quality is typically defined in terms of conformance to specification, freedom of defects, and fitness for purpose [3, 8]; IEEE glossary has many definitions for software quality [7]:

- The totality of features and characteristics of a software product that bear on its ability to satisfy given needs.
- The degree to which software possesses a desired combination of attributes.
- The degree to which clients or users perceive that software meets their composite expectations.
- The composite characteristics of software that determine the degree to which the software in use will meet the expectations of the client.

According to the previous definitions, a software product quality is basically measured by the degree to which the specified software accomplishes clients' expectations and desired functions, any software product that lacks those features is considered to have poor quality, this matches the findings in our research as 84% of respondents considered post delivery modifications as indication of poor product quality.

To guarantee high quality software, attention should be paid to the quality of development process itself, quality assurance procedures should be applied to monitor development activities as well as their outcomes. For any quality assurance procedure, two questions need to be investigated on a regular basis [13]:

- Is the right system being build?
- Is the system being build correctly?

These questions take us back to requirements. In order to develop a high-quality system that fulfills client needs, the right client requirements should be specified, this can't be achieved unless an interactive requirements management procedure is used to continuously refine and insure the quality of requirements list through the elicitation, specification and validation process, mainly more efforts should be put to improve requirements elicitation in order to handle the changing nature of requirements [1, 5], this requires using suitable methods for an iterative elicitation process. In our research, 50% of respondents stated that using iterative method like prototyping helped them better understand their client needs and manage the development process, as the clients witnessed the information they provided revolving into a working product, they were more excited about contributing in the development process. On the other hand, the 'knowledge diffusion' created by the prototype helped clients repair any wrong or inappropriate requirement they had provided previously or provide any missing ones. Moreover, implementing iterative elicitation process improved product quality as it minimized product modifications, especially after delivery modifications, which made the development process more profitable for both clients and development companies.

Conclusion

Software quality is a critical issue in software development; many systems failed or were evaluated to have poor quality as they don’t provide the essential business functions. The quality problem occurs due to the defect in the requirement management process, a big gap exists between theory and practice in requirements area because it's usually considered to be less important than other development activities. Problems in requirements occur either because the elicitation process is not a systematic one or because the standardized process is not performed correctly, which will cause the wrong requirements to be elicited and the wrong product to be built.

Ideal elicitation process should deal with problems of scope, communication, and requirements volatility. Implementing an iterative elicitation process on one hand can manage the changing nature of requirements and facilitate communication between the communities participating in the development process on the other hand, which minimizes requirements errors and improves the overall software quality.

References:

[1] Christel, Michael G. and Kang, Kyo C., Issues in Requirements Elicitation, 1992.

[2] Etien, Anne and Salinesi, Camelli, Managing Requirements in a Co-Evolution Context, 2005

[3] Fitzpatrick, Ronan, O'Shea, Brendan and Smith, Peter, Software Quality Revisited**.**

[4] Herlea, Daniela Elena, Users' Involvement in the Requirements Engineering Process.

[5] Hickey, Ann M. and Davis, Alan M., Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes, 2002.

[6] Hochmüller, Elke, Quality Improvement Through Quality Requirements Management.

[7] Institute of Electrical and Electronics Engineers. IEEE Standard Glossary of Software Engineering Terminology.

[8] Lanman, Jeremy T., Software Quality – Measurements and Management, November 2001.

[9] Leite, Julio Cesar S P., A Survey on Requirements Analysis. Advanced Software Engineering Project Technical Report RTP-071, University of California at Irvine, Department of Information and Computer Science, June 1987.

[10] Mead, Nancy R., Requirements Engineering for Survivable Systems, September 2003.

[11] Nuseibeh, Bashar and Easterbrook, Steve, Requirements Engineering: A Roadmap

[12] Playle, Greg and Schroeder, Charles, Software Requirements Elicitation: Problems, Tools, and Techniques.

[13] Software quality paper: Requirements management potential and trends SQS software quality systems, March 2006.

[14] Toro, A. Duran, Jimenez, B. Bernardez, Cortes, A. Ruiz, and Bonilla, M. Toro., A Requirements Elicitation Approach Based in Templates and Patterns