ANALYSIS OF REQUIREMENT ENGINEERING

PROBLEMS AND PROPOSED SOLUTIONS

Dr.M.P.Singh, Rajnish Vyas

Abstract—Requirements form the basis for Project Planning, Risk Management, Acceptance Testing and Change Control. Project manager’s job is to ensure that everyone understands the scope of the project and it is synchronized with the ever-evolving requirements. As projects become more complex, this becomes more challenging. It is difficult to manage projects more effectively when requirements change occurs in all phases of software development. Requirement engineering for such projects becomes tedious. System may fail due to user rejection if the requests for requirement changes are ignored and on the other hand failure to manage requirement changes during life cycle can increase the development time and cost. What happens if requirements are not engineered properly? Will they cause any problem? Is it a technical job or a managerial job?

In this paper, we would like to summarize the most common requirements engineering problems gathered in a study conducted by us. We have also listed the negative impact of these problems and most importantly proposed some solutions that can help us in avoiding these problems or at least fix them once encountered.

Index Terms: Requirement Engineering, Requirement Problems, Negative Impact.

I. INTRODUCTION

Software development is a dynamic process where demands for changes seem to be unavoidable. The requirements form basis of project schedule, cost and project performance, thus any change in requirements during SDLC is negatively associated with software project schedule, performance and cost. Thus requirement engineering activity plays a very important role in product development. Requirements engineering is a relatively mature discipline and has many well-known methods and techniques for identifying, analyzing, specifying, managing, verifying and validating a system’s requirements. But then, why there are still so many defects in requirements specifications? Why are requirements errors still a major root cause of many project failures in terms of significant cost and schedule overruns, failures to deliver all of the functionalities specified and systems that do not have adequate quality? Do we need new and radically improved requirements engineering methods, techniques and tools? Or do we just need to put into practice what many professional requirements engineers have been recommending for years?

In this paper, we would like to summarize the most common requirements engineering problems. These observations are based on the response of respondents having experience of many years working on real projects as a project lead, requirements engineer, consultant, trainer and developer. We have also listed the negative consequences of these problems, and most importantly suggested some steps that can help us avoid these problems or at least fix them once they arise. Even though these problems are not new but can be revisited as they are very common. This is most likely because the practices recommended by professional requirements engineers have not been put into practice. The study suggests that these problems are frequently encountered and cause most damages.

II. REQUIREMENT ENGINEERING

Requirements engineering (RE) has traditionally been seen as the first part of the software life cycle. The functionality and characteristics of the system to be built must be elicited and documented in a requirements specification (RS).

In order to create some common understanding of the requirements engineering domain, the term requirement needs to be defined. The IEEE software engineering glossary [IEEE, 90] (and in a similar way also [Thayer, 90]) defines a requirement as: “(1) A condition or capability needed by a user to solve a problem or achieve an objective; (2) a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents; (3) a documented representation of a condition or capability as in (1) or (2).”

Generally speaking requirements should only state “what a system should do rather than how it should do it” [Sommerville, 1997A]. According to Alan M. Davis, 2005, Just Enough Requirements Management, Requirement can be defined as “An externally observable characteristics of a desired system”.

According to Ian Somerville, the requirements for a system are the descriptions of the services provided by the systems and its operational constraints.
Table I represents results of study conducted by Standish Group about why projects are impaired and ultimately cancelled.

<table>
<thead>
<tr>
<th>Sl.No.</th>
<th>Project Impaired Factors</th>
<th>% of Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*Incomplete Requirements</td>
<td>13.1</td>
</tr>
<tr>
<td>2</td>
<td>*Lack of User Involvement</td>
<td>12.4</td>
</tr>
<tr>
<td>3</td>
<td>*Unrealistic Expectations</td>
<td>9.9</td>
</tr>
<tr>
<td>4</td>
<td>Lack of Executive Support</td>
<td>9.3</td>
</tr>
<tr>
<td>5</td>
<td>*Changing Requirements &amp; Specifications</td>
<td>8.7</td>
</tr>
<tr>
<td>6</td>
<td>*Didn’t Need It Any Longer</td>
<td>7.5</td>
</tr>
<tr>
<td>7</td>
<td>Lack of Resources</td>
<td>10.6</td>
</tr>
<tr>
<td>8</td>
<td>Lack of Planning</td>
<td>8.1</td>
</tr>
<tr>
<td>9</td>
<td>Lack of IT Management</td>
<td>6.2</td>
</tr>
<tr>
<td>10</td>
<td>Technology Illiteracy</td>
<td>4.3</td>
</tr>
<tr>
<td>11</td>
<td>Other</td>
<td>9.9</td>
</tr>
</tbody>
</table>

Table I: Projects Impaired due to requirement errors (Those marked with an asterisk are related to requirement.)

III. DIFFICULTIES IN REQUIREMENTS, THEIR NEGATIVE IMPACT AND SUGGESTED SOLUTIONS

A. DIFFICULTIES IN REQUIREMENTS

The following are some of the most important difficulties associated with requirements engineering practices:

1) Poor Requirements Quality

The actual quality of many specified requirements is poor. By poor requirements quality, we mean that many ‘requirements’ specified in requirements specifications are ambiguous, incomplete, inconsistent, incorrect, obsolete, specified using technical jargon rather than the terminology of the user or business/application domain, infeasible to implement, not actually mandatory (i.e., merely nice-to-haves on someone’s wish list), irrelevant to the system, lacking in necessary metadata such as priority and status, untraceable, in a form that is unusable to the requirements many stakeholders, unverifiable and unvalidatable [Firesmith 2003]. This problem occurs because many requirements engineers are inadequately trained. Other major causes of this problem are the myths that it is costly, difficult and impossible to produce good requirements.

2) Requirements Not Traced

Although the importance of requirements tracing is widely recognized, mentioned in the agreements and is an integral part of most of the requirements engineering methods and training classes, many requirements could still not be properly traced. The documentation of the requirement sources is not done. The requirements are neither assigned to the architecture and design nor to the test sets that are used for verification.

In complex projects it is impossible to make requirement tracing manually due to the large number of requirements. It is also difficult and resource-intensive to perform this even with the help of the available tools.

3) Missing Requirements

Midsize projects frequently have hundreds of requirements and many large projects may end up with thousands of requirements. It is very common that some important functionality is skipped during the requirements gathering. These minor slips do not usually cause much harm so long as the omissions are identified and added to later releases.

The real problem is that many architecturally-significant requirements are accidentally overlooked. These requirements are usually nonfunctional quality requirements which specify the acceptable standards of quality i.e. availability, performance, reliability, safety etc. This usually happens because of the presumption of the stakeholders (who are the main source of requirements specification) that these requirements are obvious to be included and need not be explicitly mentioned.

4) Excessive Change in Requirements (Requirements Volatility)

As most systems have long development cycles and lifecycles, it is obvious that requirements will change during the lifecycle. They will change, it is effectively impossible to freeze requirements in practice. Systems have to evolve as business needs change (e.g., with the advent of new competitors and new government regulations). The need for continuous change in requirements is the foremost cause why industries prefer to use iterative, incremental and parallel development and life cycles. But changing a system’s requirements to meet the system’s stakeholders’ changing needs has its own problems. Stakeholders will want to frequently add a few new requirements here and change one or two existing requirements there. But when this happens in an uncontrolled mode, we get the perennial problems of excessive requirements volatility and scope creep.

5) Insufficient Verification of Quality of Requirements

This problem is not about the verifying whether the built system implements its requirements. Rather, it is about verifying early in the development process whether the requirements are quality requirements and avoid the negative implications resulting from poor requirements.

Figure 1: Missing Requirements identified at later stages
Usually the requirements are verified informally with the colleagues or as an effect of the major stakeholder’s review. While both reviews are somewhat helpful, they have not proven effective in identifying requirements defects.

6) Inadequate Requirements Validation

A major task of requirements engineering is to have the stakeholders validate their requirements to ensure that the requirements completely and correctly specify their needs. Unfortunately, requirements are not always properly validated by their stakeholders. One main cause of this could be that requirements engineers usually do not have adequate access to stakeholder representatives. This could be on contractual projects and there are procedural limitations on the availability of stakeholders to validate the system requirements. Other important cause of this may be that the project’s requirements engineering method may not include requirements validation, perhaps due to ignorance of the tasks comprising requirements engineering or a lack of resources to properly perform all of the requirements engineering tasks. Sometimes requirements validation is dropped due to a lack of stakeholder time, project schedule or project funding.

7) Improper Requirements Management

Many projects do not properly manage their requirements. They document their requirements in paper or in spreadsheets instead of creating a single pool. Different kinds of requirements are also stored separately in different documents. Thus any change in requirements could not be properly reflected in all the related/recursive requirements.

8) Poor Requirements Process

In many projects, the requirements elicitation is mostly un-documented. It is usually incomplete in terms of either missing or inadequately documenting important tasks, techniques, roles and work products. The requirements engineering method is often in appropriate. Instead of having requirements engineering method that include appropriate techniques for engineering functional, interface, data and quality requirements as well as for mandated constraints any one method is used to engineer all type of requirements. The method used is often chosen because it was used more or less successfully once before by a member of the requirements team rather than because it is appropriate for the engineering of the requirements of the specific system to be developed or updated.

9) Inadequate Tool Support

Requirements engineers usually do not have or do not use tools while engineering the requirements. For example many requirement engineers create their requirement repository using requirement specification document, others use word files, excel sheets or may be some databases tables. Although some good tools such as CaliberRM from Borland, RequisitePro by IBM, or Telelogic’s DOORS are available but being used by few requirement engineers for managing requirements.

Diagrams (e.g., UML use case diagrams, sequence/swimlane diagrams and state charts) are a major part of most requirements models and they should be documented for later use. When it comes to requirements identification and analysis, many requirements engineers use simple drawing tools while others use CASE tools (e.g., IBM/Rational Rose or Telelogic’s Rhapsody) to draw diagrams.

10) Inexperienced/ Untrained Requirements Engineers

Certain managers assume that because requirements are usually specified using languages such as English, so any reasonably literate person should be able to talk to a few stakeholders and write down what they want. Also, it is believed that domain experts (business analysts and business experts) who understand the application domain, but who know nothing about requirements engineering can also become requirements engineers. While these two beliefs are untrue, it is not uncommon to see such people involved into the position of requirements engineer without having formal training and experience in requirements engineering. Requirements engineering is often a position that is little valued by technical people, who do not understand that it is an engineering discipline in its own with its own methods, techniques and tools. In fact, being a good requirements engineer requires some of the same characteristics of a good architect.

Both are needed to have a big-picture viewpoint and should be capable of communicating with technical and non-technical people. Usually this is presumed that there are very few prospects for career advancement of requirements engineers. In general most technical people consider the role to be closer to that of manager than technologist.

B. NEGATIVE IMPACTS

1) Poor Requirements Quality

Requirements engineering is the first step of development process in which major mistakes can be made. The negative impacts of this are felt in later stages of the development such as design, implementation and testing. Poor-quality requirements greatly increase development and sustainment costs and often cause major schedule overruns. Defects discovered once a system is fielded cost 50 to 200 times as much to correct as they would have had they been found during requirements evaluations [Boehm and Papaccio 1988], and these depressing figures have not changed significantly since then. Also discussed in [Wiegers 2001],
“Industry data suggests that approximately 50 percent of product defects originate in the requirements. Perhaps 80 percent of the rework effort on a development project can be traced to requirements defects.”

2) Requirements Not Traced
Lack of tracing makes it difficult to know the impact of proposed and actual changes, both to the requirements themselves and the architecture, design and implementations derived from them. When changes occur as they will on any real venture the requirements and both the upstream and downstream work products get out of sync as inconsistencies develop among them. It becomes more difficult, expensive and time consuming to perform designing, implementing and testing.

3) Missing Requirements
Missing requirements are much harder to spot during requirements evaluation than incorrect or poorly-stated requirements, their absence is often missed until the system is integrated, undergoing operational testing being manufactured or being deployed. In the worst case the missing requirements may not be discovered until the system is in use by hundreds, thousands or even an even larger number of users. Such requirements are typically much more difficult and expensive to fix at these stages. For example, it is often difficult to add on performance, reliability, safety, and security to an existing product. Missing requirements is the cause of major system failures and accidents.

4) Excessive Change in Requirements (Requirements Volatility)
Unmanaged and unexpected changes to requirements can raise chaos with existing designs, implementations and testing. Without a minimum amount of stability, developers cannot do their jobs and deliver new systems. The cycle of testing and fixing the defects becomes never-ending. Scope creep results from more requirements instead of less. Thus, it typically significantly increases the cost and time required to build new systems or versions of existing systems. Sadly project budgets and schedules are often neither sufficiently flexible nor updated to remain consistent with the new requirements. This causes projects to rapidly go over budget and milestones to slip.

5) Insufficient Verification of Quality of Requirements
Requirements defects in the quality of requirements that are not identified during the requirements engineering process will negatively impact all subsequent activities. When eventually discovered, correction of these defects will be significantly more expensive and will take more time to fix than they would had they been found and fixed during early requirements verification.

6) Inadequate Requirements Validation
A lack of proper requirements validation with the stakeholders typically results in requirements that are incomplete because they fail to specify important stakeholder needs or they are incorrect because of misunderstandings between the requirements engineers and the stakeholders. The resulting system may then be unacceptable to stakeholders even if it has been verified by testing to meet its requirements. Fixing these problems later can have major negative impacts on cost and schedule and some functionality may be missing at the time of delivery.

7) Improper Requirements Management
Requirements stored in paper form rather than in a central requirements repository are difficult to create, manipulate and maintain. Scattered requirements are hard to find, sort, query and maintain. Also, Lack of centralized, automated management of requirements makes it difficult to capture, analyze and report requirements metrics (e.g., requirements stability, maturity, and completion). Lack of access control makes it difficult to limit access to sensitive requirements and to achieve proper change control.

8) Poor Requirements Process
The output of a poor requirement processes is a poor product. Inappropriate methods are inefficient and ineffective. When different requirements engineers and requirements engineering teams use poorly documented methods, they produce inconsistent requirements, which are difficult for designers, implementers and testers to use. Methods lacking of necessary detail cause the requirements engineers to waste time arguing over what to do and how to do it. They will also make unwarranted assumptions about how parts of the method should be performed. The use of a generic requirements engineering method often results in a mismatch with the specific needs of the project. If the generic requirements engineering method is not properly tailored or if a project-specific method is not used, then it will not produce optimal results [2].

All of these problems and associated negative impacts ultimately cause budget and schedule overruns as well as the delivery of system with missing capabilities and added defects.

9) Inadequate Tool Support
Lacking in use of adequate tools for requirement documentation and management makes these processes difficult to manage. It is extremely labor-intensive to manually produce and maintain a non-trivial amount of requirements. Most of the requirements models are not properly documented and stored. Without tool support, inconsistencies significantly increase and the documented requirements easily get out-of-date.

10) Inexperienced/ Untrained Requirements Engineers
Requirements engineers without training, expertise or motivation do not tend to understand and follow good requirements methods and therefore do not tend to produce good requirements. For such people, the job can be frustrating and a source of low morale and self-esteem. In such case the position of requirements engineer is viewed as a no-fun, dead-end job for performers. Thus, poor productivity and excessive staff turnover can result and the gathered requirements may not be sufficient and as per need.
IV. PROPOSED SUGGESTIONS

- Poor requirements quality is currently the number one problem in requirements engineering and solving it will go a long way towards improving software and system development.
- Formal training about characteristics of good requirements (including example of both good and bad requirements) must be provided to the requirements engineers, stakeholders and requirements reviewers.
- Ensure that all of the requirements have the appropriate characteristics (unambiguous, complete, correct, mandatory, readable, etc.). Simple tools should be used to identify inherently vague words being used in the requirements.
- Members of the design and testing team should be involved while verification of the quality of requirements which will ensure the feasibility and verifiability of the requirements. Ensure that the requirements engineers are enabled and required to collaborate with stakeholders until the requirements have sufficient quality. Finally, requirements engineers should rework or delete all requirements that lack the required characteristics of good requirements.
- Ensure that requirements tracing is mentioned in the contract and explicitly. Also be sure to go-ahead and verify the tracing of all requirements. Provide user friendly and scalable tool support for requirements tracing. Ensure management understands the negative impact of not tracing requirements and obtain support for proper tracing, including providing adequate resources to trace the requirements. Ensure that tracing occurs both early in the project development cycle as well as later during design, development and maintenance. Also, ensure that the evaluation of requirements tracing is a documented part of the requirements verification method.
- Requirements engineers must gather the requirements themselves instead of relying on stakeholders to put up what they want. The requirements team should collaborate with specialty engineering teams (e.g. reliability, safety, security and usability) and representatives from all groups of stakeholders when eliciting requirements.
- The primary solution is not to fix existing requirements and prohibit the addition of any new requirements. Using a lifecycle that allows requirements changes is a good idea. But changes to the requirements must be properly managed. For each release of the system, the requirements must be base lined and fixed at appropriate milestones within the development/ update cycle. Base lined requirements should be placed under configuration control like any other major work product, and the impact of changes to these requirements needs to be determined before the changes are authorized to take place. Finally, budgets and schedules need to be updated whenever there is any nontrivial change to the requirements.
- Make sure that requirements validation is a fundamental module of any requirements method. Ensure that requirements validation is included into the project’s schedule and budget as well as the schedules and budgets of the system’s stakeholders. Finally, remove unnecessary obstacles separating the stakeholders and the requirements team.
- To deal with the large number of requirements and the constant changes to them, store the requirements in a database or the repository of a requirements tool. Store the requirements models and diagrams with or linked to the requirements. Store all important attributes about a requirement (i.e. metadata) with the requirement so that they are easy to manage and maintain. Do not spread out different kinds of requirements; rather, keep them all in a central repository. Ensure that this requirements repository (and tool) supports access control, including prohibition of unauthorized access to sensitive requirements.
- Have an experienced requirement engineer to ensure that the requirements engineering method is complete, incorporating all of the important method components including tasks, techniques, roles and responsibilities and work products. The quality organization should also audit the requirements engineering process. Ensure that the method components are mature and have been successfully used on projects that were similar in size, complexity and type and that developed similar systems. Ensure that everything is properly documented and easily understood by their target audiences, and contain the appropriate level of detail based on the training and experience of the people who will use them.

Where ever possible, create a project-specific requirements engineering method that meets specific needs. We may use a commercial tool (e.g., RUP from IBM/Rational, which seems to be the most commonly mentioned process engineering tool in the software engineering community. We may consider reusing free, open source method components to construct project-specific requirements engineering method.
- Use a powerful, yet user-friendly, requirements management tool that enables the storing of requirements metadata. Ensure that these tools support the configuration management of the requirements and their models. Where possible, choose an integrated toolset that supports traceability as well as the forward engineering of requirements through designs to implementations and tests and reverse engineering from these back to the requirements.
- Carefully select people with the right combination of training, experience, motivation, mindset and people
skills to be good requirements engineers. Provide them with significant amounts of training, including classes, conference tutorials, books and journals. Ensure that others, including both management and the technical staff, understand the importance of the role they play in success of project.

V. CONCLUSIONS

In this paper, we have briefly described the few most important problems negatively impacting the requirements for software projects. For each problem, we have described some of its major negative impact, and the most important things we can do to either evade these problems or fix them. We have also suggested several solutions. Applying these practices to solve or avoid requirement problems will help solve or prevent several other problems.

These are perennial requirement engineering problems, the suggested practices as solutions may help. Our motive is to draw the attention of requirement engineering professional towards the practices recommended by the professional requirements engineers in the past and not to develop new and improved requirements engineering methods and techniques. Although it leads to a new problem that solving these problems will take a significant amount of awareness raising, training and support of management. Important point is:

• How many of us will put these suggestions into practice?
• How many of us will continue to suffer the negative consequence if we don’t?

ACKNOWLEDGEMENT

I wish to express my sincere thanks to my guide for providing his valuable guidance while preparing the paper and during the study. I also thank him for reviewing my paper and giving suggestions for further improvement in the same.

REFERENCES


