Managing Changing Requirements

Prashasti Rikhari, Arko Bagchi

Lovely Professional University

Abstract- We cannot freeze requirements at the requirement gathering phase. Requirement changes after the requirement gathering phase because of various factors like business change, technology change, internal changes in the organization and better understanding of the problem. Managing Changing Requirements are very crucial for success of any software development and its organization. We have to manage the requirement so that we can achieve project objective as well as business objective. In this research paper, we will give a complete strategy that will work on the issues related to Managing Changing Requirements and will provide a solution to manage the Changing requirements. We will also give some precaution steps which can be used before implementing the strategy so that we can achieve requirement management in a more effective way.

Index Terms- Changing Requirements, Requirements, Requirement elicitation, Requirements Management.

I. Introduction

Requirements management for globally managed large scale projects is a complex task, primarily due to the involvement of several stakeholders, multiple technologies, enormous information, some internal changes in the organization and frequently changing business needs [1]. It has been identified that customer requests for changes in each development phase of a software project. However maximum change requests are received from customer in maintenance phase of the software project. It has been identified that if more changes are requested in early development phases, then fewer changes are requested in later development phases of a software project [2]. Managing Changing Requirements are very crucial for success of any software development and its organization. Although the analyst paid a full attention to collect the complete set of requirements, the requirements will still change after the analysis phase. If we do not manage these changing requirements we have to face various technical, economical and managerial problems.

Changing requirements is one of the major requirements related problems, and it is considered as a major cause of the failure of software projects [3, 4].

Before discussing the strategy, first I would like to discuss the various issues that are related to requirement management.

II. Issues in Requirement Management

The major consequence of requirements problems is rework—doing something again that you thought was already done. Rework can consume 30 to 50 percent of your total development cost [5]. Requirements errors account for 70 to 85 percent of the rework Cost [6]. Getting the wrong requirements and developing a wrong system degrades the quality of the system. Another problem is ambiguity. It occurs when a single requirement has different meanings to different stakeholders, it leads to wasted effort. Another problem is gold plating, which means adding unnecessary features in the software which increase the cost of the software as well as the complexity. Managing requirements communicated through mediums such as emails, chat messages, etc. is difficult due the lack of explicit context and the extensive use of unstructured information. The result of all of these kinds of requirements problems is to slow the pace of software development. It doesn't do any good to work very efficiently on the wrong requirements. If we don't get the requirements right, it doesn't matter how well our team executes the rest of the project.

In some areas for example informal information gathering, implied functionality, inadequately defined requirements, and a casual change process, problem arise due to shortcoming in the ways that people acquire, document, agree on, and modify the product's requirements.

Therefore if we do not manage these changing requirements, we have to face various problems, for example-economic problems which are related to cost and schedule of a project, technical problems which are related to the defects, bugs and the low quality software, managerial problems which are related to the effect of technical and economical aspects on the sponsors and customers.

III. Significance of Managing Changing Requirements

People often think that the changing requirements have bad impact to the software development process. The software development team needs to look at the optimistic part of changing requirements as it facilitates to reduce the risk of project failure and increase the quality of the resulting product. Getting the wrong requirements and developing a wrong system is the biggest risk to the project failure [7].

If we have a good quality software, than it will contain a very few bugs and errors which increase the customer satisfaction. If we manage requirements according to the market that the software will definitely compete well in the business which result the business success of the software. Adoption of latest technology will increase the success of the project. It also reduces the chance of system rejection upon rollout.

Therefore we must need a strategy, which will solve all the issues of requirement management and will provide a solution for it. Before I give the strategy for managing changing requirements, I want to give some precaution step before implementing this methodology on any software, so that we can achieve requirement management in more effective way.

IV. Precaution Steps

Precaution steps for managing changing requirement are given below-

4.1 Firstly we need to take care in requirement elicitation phase. We must gather only those requirements which are unambiguous, logically consistent, relevant, clear, can be completed within the available resources, realistic and improve the quality of the software. So that the requirements which does not fulfils these constraints cannot harm the software and development process in future.

4.2 During the development of the software, the development team must continuously interact with the stakeholders. In most of the organizations, during software development process, the development team interacts with the stakeholder only in requirement analysis phase. This should not be the case. The development team should ask the stakeholders about their problems and new ideas. It can be achieved by showing them the prototypes of the system. If we spend more time on requirements engineering in the early stages of a project, it's supposed to save our time later in the project.

4.3 Requirement creep occurs, when changes in the system requested by the stakeholders are very big such that it is impossible to control the cost and schedule of the system. Therefore we must avoid the big changes just to make the customer happy. We can add those extra features in other releases of the software.

4.4 There must be a way to maintain the requirement document. Every time when a new requirement is added or if there is any change in the requirement we must keep track of these things. It can be accomplished by giving a new version number to the requirement document. The changes can be highlighted in the requirement document. We can also store the old version of the documents for future analysis.

V. Strategy

This strategy consist of 8 steps, which are given below

5.1 First Step (Validating the new requirement)

First thing that we need to consider is look at the optimistic part of changing requirements. It means if we are not ready for implementing the changing requirements than for sure we cannot implement the changing requirements.

If from the start of the software development, we have no interest in working with these changing requirements, than at the latter stage of software development it will create a big problem because of our carelessness at the initial stages.

We have to set our mind for implementing these changing requirements. We need to consider that these changes are necessary and good for improving the quality of the software which will save our software from the biggest failure that is developing wrong software with wrong requirements. Therefore in first step we receive the changing requirements from any of the stakeholders.

5.2 Second Step (Appropriate classification of the requirement)

In second step we classify it according to its area. For example-Functional Requirement, Non Functional Requirement, Platform Requirement, Hardware Interface Requirement, Software Interface Requirement, External Interface Requirement, User Interface Requirement, Communication Requirement. It will help in clearly identify the different feature and properties of the proposed requirement. It also avoid various misunderstanding related to the particular requirement.

5.3 Third Step (Traceability)

In third step we perform traceability. That is we identify that if the proposed requirement is conflicting with other requirements. For example- if the new requirement is related to increasing the reliability of the software, than it will conflict with the complexity and usability requirement of the software.
Traceability matrix can help in tracing the requirement. It is used to verify that all requirements are met and to identify changes to the scope when they occur. Before implementing any new requirement or making changes in the existing requirement, we must have complete knowledge of the already existed requirements, so that we can find out the relation between the new or already existed requirements. Sometime, the new requirement can be added only when its previous related requirement is completed, traceability matrix gives us all these information regarding the already existed requirement.

Table 1: Traceability Matrix

Req#	Name	RFP#	DDD#	PPT#	TS#	Verification
1	Calculate DOB	CGA 001	DDD 001	3.2.3 4.1..2	TS 001 TS 015	Yes/NO
2						

Req#: Requirement Number; for each project requirement, begin to list them on the RTM in a numerical order and group them by function.

Name: Enter the name and brief description of the requirement

RFP #: Request For Proposal (RFP); specify the identification number of the requirement as listed in the RFP.

DDD #: Deliverable Definition Document (Also referred to as the Deliverable Expectation Document- DED); use the RFP requirement number as a reference for the DDD that is created for the requirement.

PPT #: List the MS Project Subtask and Task numbers that are associated with the requirement.

TS #: Test scripts should be prepared for the actual testing process.

Verification: Indicate the completion of the signoff process [8].

5.4 Fourth Step (Review of the requirement)

This step includes the review of the proposed requirement. All the stakeholders (end users, customers acquiring the product, requirements analysts, project managers, developers, testers, regulators and auditors, manufacturing staff, sales and marketing, and field support or help desk staff) are involved in the review. They give their opinion for the proposed requirement. They also can accept or reject the proposed requirement. If any stakeholder reject the requirement than the owner of the requirement must modify his/her requirement according to the review of the stakeholder.

5.5 Fifth Step (Evaluation of the requirement)

A separate team which can be considered as the evaluation team for the proposed requirement evaluate all the reviews that are collected from all the stakeholders. Therefore In this step impact analysis is performed by the evolution team. The team evaluate the reviews on the basis of cost, schedule, risks, functionality, customer and external stakeholders. It will be accepted if, it clears the cost benefit analysis test, does not increase the schedule of the project to a uncontrollable range, does not conflict with other requirements, can be expressed in a clear and consistent notation, must be unambiguous, not over-constrain the design of the software, must be logical consistent, relevant and clear, must be achievable with the available resources, must be verifiable, it means there must be a way to test it for ensuring that it is correctly implemented, does not introduce the high level risks to the software for example- software failure, increase the functionality of the software or quality of the software and satisfy all the stakeholders, otherwise the team reject the proposed requirement.

5.6 Sixth Step (Documentation of the requirement)

According to evaluation result, if the proposed requirement is accepted than the owner of the requirement baseline the requirements. And it is documented and placed in subversion.
Than all the parties affected by the new proposed requirement are informed, it is accomplish by the collaborative power of the Web to make requirements available to the entire team whenever and wherever they need them – not rely on a few individuals to remember to email out updates every time a change is made. And if the proposed requirement is rejected than the reason for disapproval of requirement are forwarded to the owner of the proposed requirement.

5.7 Seventh Step (Adjusting different activities)

At last we need to adjust the different activities or plan which reduce risks according to the new proposed requirement. For example- for a new programming language, we schedule training courses and time for the programmers to practice their new programming skills.

5.8 Eighth Step (Acceptance testing)

After modification the new copies of the software is released for user acceptance testing. And after verification and validation the master copies of the document will be replaced by the new one.

VI. Future Work

This strategy can be applied to any real life project, so that its efficiency can be measured. It will help a requirement management tool for managing the changing requirement.

VII. Conclusion

This Strategy is a real time process. It can be applied into any development process and any of the phases of software development life cycle. Project plans and commitments therefore need to anticipate change, rather than expecting that an initial stage of gathering requirements is all that's needed.

Acknowledgement

We thank our faculty members and friends from the bottom of our heart for their time and patience in participating in our experimental study.

References

[1] S.Ghosh, A.Dubey and S.Ramaswamy, "C-FaRM: A Collaborative and Context Aware Framework for Requirements Management", 2011.

[2] M.W.Bhatti, F.Hayat, N.Ehsan, S.Ahmed, A.Ishaque, and Z.S.Sarwar, "An investigation of changing requirements with respect to development phases of a software project", 2010.

[3] S.Lock and G.Kotonya, "An integrated framework for requirement change impact analysis", 1990.

[4] E.Oz, "When Professional Standards are Lax, The CONFIRM failure and its Lessons", 1994.

[5] B.W.Boehm and P.N.Papaccio, "Understanding and Controlling Software Costs", Volume 14 Issue 10, October 1988.

[6] D.Leffingwell, "Calculating the Return on Investment from More Effective Requirements Management", American Programmer 10(4):13–16, 1997.

[7] S.A.Perumal and G.Kavitha, "Changing Requirements – Correlated to Risk or Quality?", Vol.3, No.1, February 2011.

[8] T.Carlos, "Requirements Traceability Matrix", 2008.

Authors

First Author – Prashasti Rikhari, B.Tech-M.Tech (CSE), Lovely Professional University, prashasti29@gmail.com
Second Author – Arko Bagchi, Assistant Professor, Lovely Professional University, arkobagchi@gmail.com