Development of a Structured Framework to Minimize Impact of Requirement Volatility

Harsh Dev
Professor,
Deptt. of CS&E,
PSIT, Kanpur

Ranjana Awasthi
Research Scholar
Deptt.of CS&E,
Singhania University, Rajasthan

ABSTRACT
Minimizing the impact of the requirement volatility in software development process is a critical issue for both researchers and practitioners. Requirement volatility is unavoidable; therefore we need to find correct solution to manage changing requirements. Despite having many methods and tools, that are available to manage the requirement change, we need to evolve a method that deals with the change in a way that minimizes the impact to the stakeholders. In this paper, we have proposed RVMIN framework that is incorporated in SDLC, in a way that it minimizes impact of change on the stakeholder in terms of time and cost. This framework is based on both an empirical study that we have conducted and our extensive literature review of Software Process Improvement (SPI) and Requirement Engineering (RE).

Keywords
SPI, Requirement Engineering, Requirement Volatility, RRC(Request Review Committee), RFC(Request for change), CRI(Change Request Implementation), RVMIN Framework.

1. INTRODUCTION
A Fortune 100 company embarked on a project to design and build a sophisticated software package that it would ultimately deploy to its offices throughout the world. Two years and about $10 million later, the field offices refused to use the software because it didn't do what it was intended to do. Instead of helping to streamline an important business process, the software actually hindered it [1].

According to a survey by the Standish Group [1] which had the total sample size of 365 respondents and represented 8,380 applications:

- 31 percent of all software projects are canceled before they are completed (a waste of $81 billion).
- 52.7 percent of projects cost 189 percent of their original estimate.
- In large companies, 9 percent of projects are on time and within budget.
- In small companies, 16 percent of projects are on time and within budget.

Top three reasons why projects are "impaired", according to above said survey are listed in Table 1.

Table 1. Top three project impairment factors (Standish Group survey)

Project Impairment Factors	% of Responses
Lack of user input	12.8%
Incomplete requirements & specificaions	12.3%
Changing requirements & specifications	11.8%

As this table shows, poor requirements are the biggest problem. If it isn't clear what we are to build, how can we estimate the cost of building it? How can we create a project plan, assign resources, design system?[1] We need accurate requirements to perform all these activities. Requirements evolve as project proceeds, but carefully written requirements give a basic start point. Then, as the project advances, we can fill in the details and update planning documents as the requirements grow.

Changes to software systems are unavoidable since it is not possible to come up with a complete and correct set of requirements that remains constant throughout a software system's life. Thus, the management of change in software systems has been a continual problem in the software industry. If not handled properly, it is very likely that requests for changes in requirements will have a negative impact on software quality, cause cost overruns, delays, and unsatisfied users and, in worst situations, cancelled projects.

2. SYSTEM DEVELOPMENT LIFE CYCLE
System development operates in acyclic manner beginning with the identification of user's needs, feasibility study, followed by the evaluation and cost benefit analysis of the candidate system and finally design and implementation of chosen candidate system.

The Waterfall Model is the earliest and most used method of structured system development. The water fall model is designed to follow the above said set of actives in a sequential order. Starting with system requirement engineering the model ends with system operation. This model clearly explains what activities take place in each phase.

The waterfall model is an attempt to put discipline into software development process by forcing standard documentation. Before we move to the coding phase, the design documentation has to be written. Each module has to be tested before going to the next sequent module. The typical programmer would prefer the coding before the documenting. Consequently some programmers consider the whole model to be painful, because it tries to force a discipline.

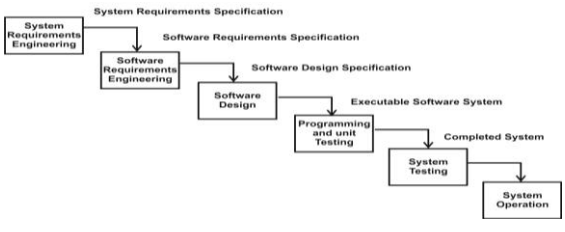

Fig 1: Classical waterfall model [Ref: Ian Somerville]

Despite many advantages of waterfall model, the assumption that is usually invalid in a waterfall process is that the requirements will not change during the lifecycle of the project. In reality, requirements change a lot in most (though not all) projects, especially once the customer gets their hands on it. The failure of traditional waterfall process to recognize this is a fundamental flaw. A mistake in the requirements phase cannot be detected in a waterfall process until near the end, when the customer gets to see the (nearly finished) product. This leads to a huge cost in correcting the mistake. Waterfall model is based on the empirical observation of 30 years ago (ref: Barry Boehm, Software Engineering Economics, Prentice Hall, 1981.) that the cost of change rises exponentially (base 10) by phases. The conclusion is that we should make the big decisions up front, because changing them is so expensive.

The traditional waterfall software process model has largely been replaced by iterative, incremental and agile approaches to software development, in order to accommodate requirements changes during the project lifecycle. As the waterfall model has been used from years to produce quality products, thus our research is based on suggesting some changes in this basic model.

3. REQUIREMENT AND REQUIREMENT VOLATILITY

Most requirements are elicited during the early stages of a software project and evolve throughout the system's life cycle. Requirements evolve or change in order to satisfy the changing needs of the system stakeholders. According to Humphrey [4], the customer does not initially know what is needed of a software system and as a consequence, requirements are wrong and will change and evolve over time.

Some relevant examples of changes, as mentioned below, are noted in [5], [6]:

•*changes in technology, which is unavoidable*
•*the requirements changes as a result of increased understanding of the problem during development*
•*the user's needs evolve as a consequence of changes in business policies and procedures*
•*the problem the system is intended solve, changes as a consequence of changes in business policies and processes*
•*market changes, and*
•*legislative or regulatory changes*

These types of changes will have more or less severe impact on the software depending on a number of factors like time of change (early/late development stages, after delivery, etc.) or type of change. Requirement changes may occur during 1) Software design 2) Coding 3) Testing 4) Implementation 5) Documentation. From the literature study as indicated above, **we can broadly specify that each requirement change is regarded as being**

1) **Either due to defect in original requirement, or**
2) **Caused due to change in requirements at later stages.**

For condition 1, we have already suggested a solution [7][8]**. In this paper we propose a framework for the cases when requirement volatility is caused due change in requirements at later stages i.e. condition 2.** The proposed framework will not only help the requirement engineer to develop an understanding of the requirement change process, but also help in identifying & minimizing the variables that have direct effect on change, thus minimizing the impact of requirement volatility in development process.

4. THE FRAMEWORK (RVMIN): (Tool Suggested to minimize volatility) - A Basic Design process

A. Method of study

We performed the study on software developers/requirement analyst of ISO/ CMM certified companies. Main target of study was to identify the problems faced by the stakeholders (developer, requirement analyst and end users). In this study they were asked to fill in a questionnaire consisting of 35 questions. Along with some fundamental questions, the main points of study were as follows:

- What procedures are currently followed by the analysts for requirement gathering?
- How are changed requirements incorporated in the system in current scenario?
- What is the impact of requirement changes in software development process?
- What can be done to minimize the impact requirement changes software development process?

The study revealed two basic things:

1. ***Very few requirement engineers are formally trained in requirement engineering. So, this was one of the important points that were taken care of while designing the framework.***
2. ***The other point of study was that, the changes are implemented according to change management process, but some standard method needs to be designed that would reduce the impact of requirement volatility.***

Throughout the design process, we kept both these points in mind.

B. Design

The base of design of the framework was the concept that in a change management process, *if standardized methods and procedures are used for efficient and prompt handling of all changes, it minimizes the impact of change*, and consequently improves the day-to-day operations of the organization. (Wikipedia, 2010) According to ISO 20000-1 Requirements Summary, the objective of change management (9.2) is as follows:

To ensure all changes are assessed, approved, implemented and reviewed in a controlled manner.

Proposed framework/solution

The paper proposes a framework that is based on the above definition and talks about slight additions to the classical waterfall model being used today (Fig 1).

RVMIN framework has much in common with the current methodologies being used, but there are important modifications. The study conducted by us indicates that although most of the development organizations are following any one formal development model, but it was observed that in most of the organizations requirement gathering is done by experienced person, but they are most of the times not formally trained in requirement engineering. The study thus indicates two changes that are desired in the classical waterfall model [Fig 1]:

1. Addition of one mandatory step **"Strict Adherence to RE practices"** during analysis phase that incorporates use of formal training/ procedures being followed by the person doing requirement analysis.

2. Use of **RVMIN**, suggested framework, for incorporating any requirement changes that may occur at later stages.

Classical waterfall model may be modified as below:

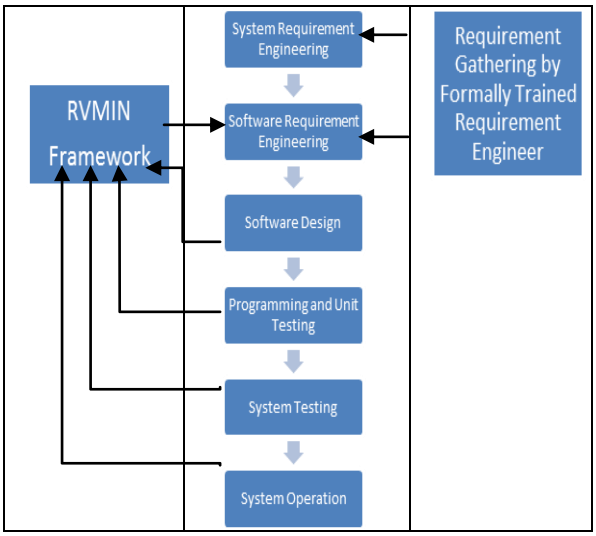

Fig 2: Modified Classical waterfall model

As seen above, requirement changes may occur during 1) Software design 2) Coding 3) Testing 4) Implementation 5) Documentation. Also, we have seen *if standardized methods and procedures are used for efficient and prompt handling of all changes; it minimizes the impact of change.* Keeping these points in mind we have designed the solution. Any change desired should pass through RVMIN Framework so that the changes are implemented in a controlled manner and impact of changes is minimized

C. RVMIN Framework

RVMIN framework suggests a wider scope for the requirement analyst by concentrating on formal procedures during the requirement change. These steps offer a way that help analyst to capture and implement the change in requirements in such a manner that impact of requirement volatility is minimized. This paper also presents the suggestions that will help the analyst to have better requirements.

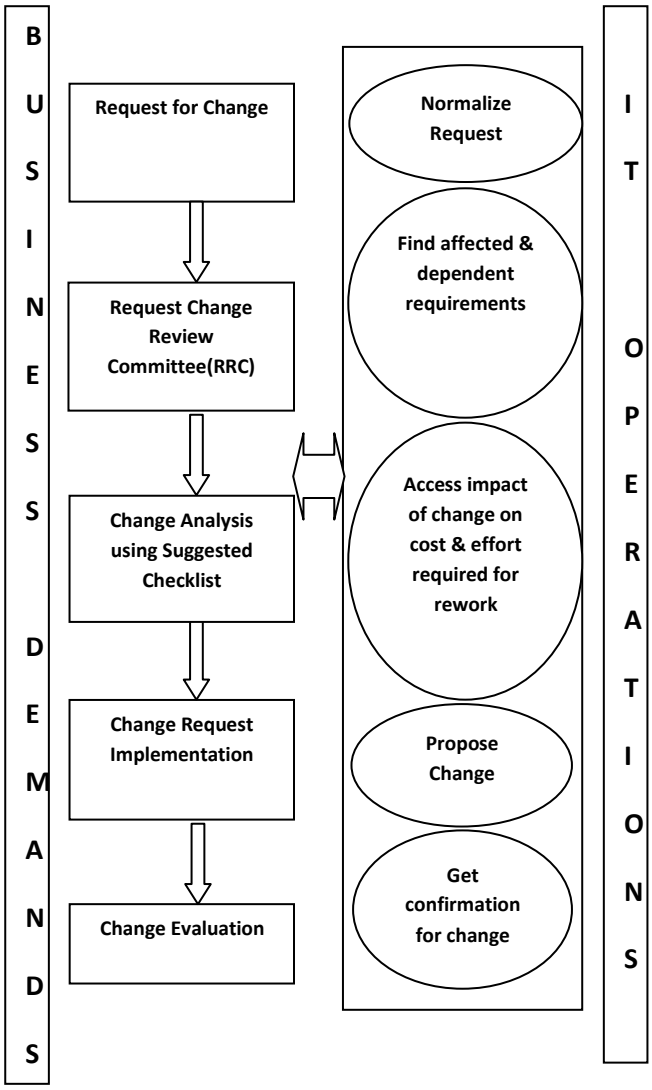

Fig 3: RVMIN Framework

The proposed framework is in accordance with ISO 20000-1 Requirements Summary. RVMIN Framework consists of four phases:

1. **Request For Change(RFC)**
2. **Request Change Review Committee(RRC)**
3. **Change Analysis using suggested checklist**
4. **Change Request Implementation**
5. **Change Evaluation**

4.1 Request for Change (RFC)

Request for change is a formal request, in writing, that is created when customer desires any new functionality in the system and formulates a REQUIREMENT. It mainly identifies the potential change desired in the system. The change request is checked for validity. Customers can misunderstand requirements and suggest unnecessary changes. This request for change may be generated, whenever the customer proposes a new change, because of any of the reasons mentioned above in introduction.

Formally, changes to a single requirement may have rippling effect throughout the system and impact on other requirements and broader organizational goals. Once the requirements are base-lined, any proposed change must be made via a formal request which is reviewed in relation to

possible impact on scope, schedule, cost, quality of product and work in progress.

4.2 Request Change Review Committee(RRC)

Requirements change review committee(RRC) should be formed having the members from both the sides. This committee will decide the degree of change and its impact cost wise and time wise and revision of PERT chart etc.
Request for change is handed over to RRC for detailed analysis. As this committee contains the members from both the sides, any implications of cost, time quality etc. can be analyzed and decision can be reached whether a change is to be implemented or not. The changes that are suggested for incorporating are to be signed by all the members.

Signing this document will ensure that both the parties are in agreement for additional cost for change in requirement. In many situations this will help in reducing the requirements volatility because customer will seriously give the requirements as additional cost is involved in requirements change. Instead of giving futile requirements, only the genuine requirement changes will be given in request for change.

4.3 Change Analysis using suggested checklist

Change Analysis using checklist suggested by us, is the most important step in this framework where the project manager determines the feasibility of request for change. A checklist is proposed to standardize the procedure, and will help the project manager to perform this activity.

Change analysis is performed in following sub-steps:

Normalize request

Most of the requests for change do not contain quality requirements. We need to improve the quality of requirements and validate them. This step may help in reducing the impact of the requirement volatility [7].

Find affected and dependent requirements

Due to the dependencies between requirements, the change of some requirements would have an impact on correlated requirements, which leads easily to the diffusion of impact. This makes the evaluation of impact uncertain and difficult.

The requirements which are directly affected by the change are discovered. Traceability information is used to find dependent requirements affected by the change.

Assess impact of change on cost & effort required for rework

If requirement changes are desired during SDLC - there are many ways to handle these changes. Two variables that play an important role in assessing the impact of these changes are:

Efforts - Person Hours (i.e. Efforts involve in design, coding, testing, etc. (i.e. COST))
Delivery Date - Time involved

Example:
1. No change in efforts / No change in Time line (i.e. delivery date)
- Accommodate changes.

2. No change in efforts / Change in Time line (i.e. more time required)
- Notify customer with new delivery date and if they are ok then accommodate changes.

3. Change in Effort / No change in timeline (with some extra effort, same time line can be achieved)
(a) If effort is not significant (example: In 200 person hour's project, extra 10 hrs. required)
- Accommodate changes

(b) If effort is significant (example: In 200 person hours project, extra 50 hrs. required) -
- Prepare revised SOW (statement of work) and send to customer for sign off
- If signed off, Accommodate changes

4. Change in Effort / Change in timeline
If effort is significant and time line also impacted with change
- Prepare revised SOW and send to customer for sign off,
- Notify customer with new delivery date. (Is it closer to deployment in production date then suggest to release in next release cycle?)
- If signed off, accommodate changes and release in next release cycle

To analyze the impact of the change on cost & effort required for rework, a checklist as proposed by Karl Wiegers is used.

Propose change

The actual changes which must be made to the requirements are proposed.

Get confirmation for change

Negotiations with customers are held to check if the costs of the proposed changes are acceptable.

4.4 Change Request Implementation (CRI)

During change request implementation, changes requested in request for change are implemented and testing and verification is carried out. Also, this is integrated to system & deployed to user site.

4.5 Change evaluation

During change evaluation, change implementation results are evaluated, and traceability links are updated.

Change request is rejected
- If the change request is invalid. This normally arises if a customer has misunderstood something about the requirements and proposed a change which isn't necessary.
- If the change request results in consequential changes which are unacceptable to the user.
- If the cost of implementing the change is too high or takes too long.

5. VALIDATION OF THE FRAMEWORK

The suggested framework, described in this paper, was validated by implementing it in real life projects to find out whether it provides any benefits to the users or not? The objective of this case study is to investigate the amount of help that RVMIN framework offers in a real project by comparing the case study that uses the framework with some other project that does not use the framework.

We chose Think Computers as the software organization where the study could be conducted. One of the main reasons for the choice was due to most of RE process had been ad hoc in most previous software projects.

Think Computers is an interdisciplinary, professional consulting firm, which has its primary purpose as the application of Managerial, Information system and Engineering skills to the solution of a wide number of problems in various commercial environments. The firm has in the past provided Software solutions and government training in size from small private organizations to ISO/ CMM companies. Owing to the requirements of professional excellence, more than half of Think computers, Technical and Management personnel have significant levels of direct industry experience in addition to academic and consulting credentials

Think Computers has a large number of developers with experience in development tools. A team of senior consultants and 4th generation developers develops the product. To provide a broad knowledge base of expertise to their clients, think Computers project teams also include specialized staff members in required disciplines. A team of senior consultants and 4th generation developers work together in developing the products.

An experimental tryouts and statistical analyses at a large scale with typical representative samples may be needed to standardize the framework. More developmental activities using the framework may be carried out by the researchers and practitioners.

5.1 Result Analysis

5.1.1 Quantitative Analysis

We chose two projects X and Y, with similar project attributes, of the company to compare the results of the study. The two projects were also similar in the number of person involved, level of complexity, allotment of resources and use of technology (VB.NET in this case) and the planned duration of the project.

Project X is developed using modified waterfall model suggested by us that uses RVMIN framework, whereas project Y used ad-hoc RE practices. It was observed that though requirements for project X were 20% more than the requirements in project Y, it required less development time to develop project X. Also, project X was only 10% overtime against 25% of project Y.

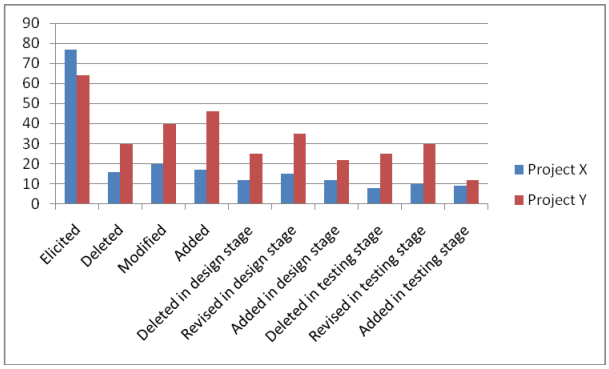

Fig 4: Quantitative Analysis of the effect of implementation of RVMIN Framework

From the above we may conclude that, the comparative evaluation of the two projects shows the advantages of using the proposed RVMIN framework.

5.1.2 Qualitative Analysis

In addition to the quantitative analysis presented in the last section, a questionnaire was conducted among all the developers, requirements engineers as well as managers who were involved in the project X. The objective of the survey was to get further feedback about the usage of RVMIN framework. The questions used in the survey are shown in Table 6.1.

Total of 4, 20 and 4 questionnaires were issued to Managers, Developers and Requirement Engineers respectively out of which 3, 14 and 4 were filled and returned.

Response of developers was also positive as more than 60% of the developers agree that the use of RVMIN framework has helped in minimizing the impact of requirement volatility.

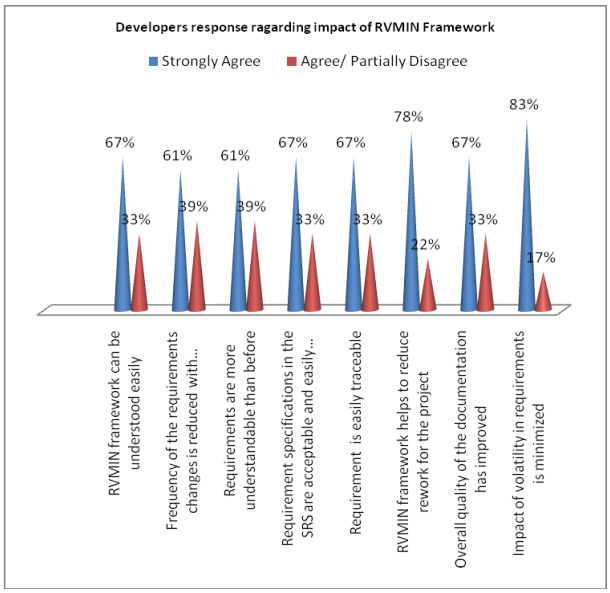

Fig 5: Response of developers regarding impact of RVMIN framework

Following figure shows response of the managers regarding questions asked about the use of RVMIN framework.

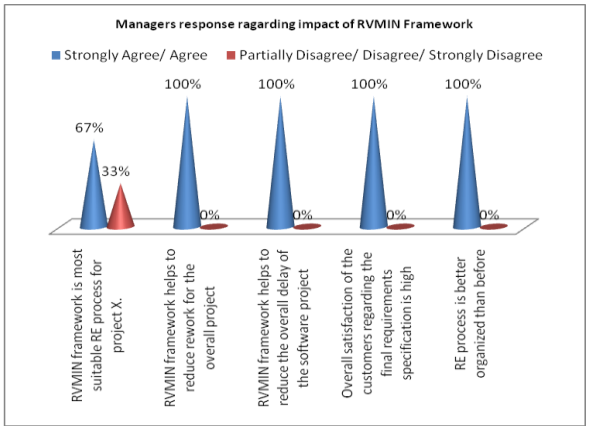

Fig 6: Response of manager regarding impact of RVMIN framework

As can be seen from above figure, management was very positive regarding the overall performance of the use of RVMIN framework for effective use of RE Process.

Same is the case with requirements engineers. This is an indication that the requirements engineers liked the idea of developing a project-specific RE process.

From Fig 7, we see that Requirement Engineers support the fact that the framework RVMIN is very helpful for developing the most suitable process model for the project.

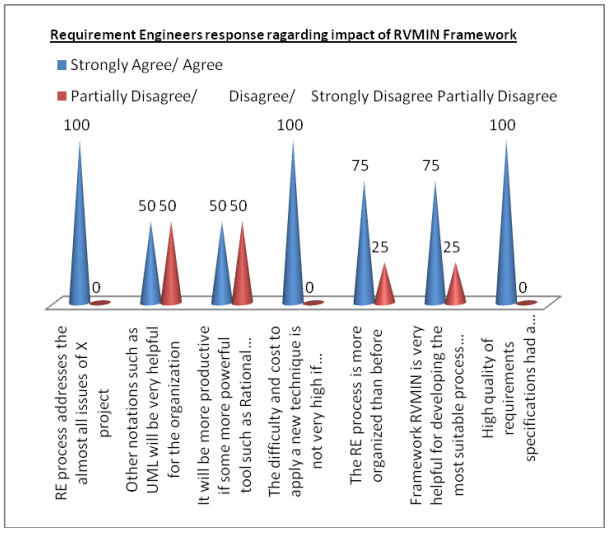

Fig 7: Response of Requirement Engineer regarding impact of RVMIN framework

The results obtained in this case cannot necessarily be generalized and cannot guarantee that similar success would be achieved in other applications as well.

6. CONCLUSION

Our research attempts to find out a method that helps the industry to minimize the impact of requirement volatility. In this paper two modifications have been suggested in the traditional waterfall model. Firstly, the requirement gathering should be performed by formally trained requirements engineer, who strictly adheres to RE practices. Secondly, to incorporate the suggested framework (RVMIN) for implementing the requirement change process, irrespective of the stage at which the change is being requested. By using these steps the impact of requirement volatility can be minimized.

The quantitative and qualitative analysis presented above suggests that Think Computers was able to develop a much better requirement specification using suggested framework. Earlier projects(Y) used ad-hoc RE processes. The requirement engineers and project manager emphasized that the high quality of requirements specifications had a positive impact on the software project. The data collected from case study in the company shows that the impact of requirement volatility was lower in project X as compared to project Y, also requirement volatility and conflicts were greatly reduced.

Using RVMIN framework, the RE process becomes very smooth and systematic, and thus minimizes the volatility in requirement. The suggested framework (RVMIN) is used for implementing the change process, irrespective of which stage the change is being requested, and is used to minimize the impact of requirement volatility. The framework has been tested and validated in real world situations in the industry and has been found suitable for use.

7. REFERENCES

[1]www.ibm.com/developerworks/rational/library/4166.html

[2] The Standish Group, The CHAOS Report, 1995

[3] I. Jacobson, G. Booch, and J. Rumbaugh, "The Unified Process," IEEE Computer, vol. 16, pp. 96–102, 1999.

[4] Humphrey W.S. 1989. Managing The Software Process. SEI Series in Software Engineering, Addison-Wesley.

[5] Boehm Barry 1981. Software Engineering Economics. Prentice Hall.

[6] Somerville I. 2007. Software Engineering. Addison-Wesley.

[7] Ranjana Rajnish et al, "Improving Requirements Quality: An Approach to Reduce Impact of Requirements Volatility", International Journal of Emerging Technologies and Applications in Engineering Technology and Sciences (IJ-ETA-ETS) ISSN: 0974-3588

[8] Dr. Harsh Dev, Ranjana Rajnish and Rajnish Vyas, "Writing Quality Requirements (SRS): An Approach to Manage Requirements Volatility", Indian Journal of Computer Science and Engineering, June'2010, Vol 1, Issue 1, pg28-37, ISSN: 0976-5166

[9] T. Javed, M. Maqsood, and Q. Durrani, "A Study to Investigate the Impact of Requirements Instability on Software Defects", SIGSOFT Software. Eng. Notes, vol. 29(3), pp. 1 – 7, 2004.

[10] N. Nurmuliani, D. Zowghi, and S. Fowell, "Analysis of requirements volatility during software development life cycle," in Proceedings of the 2004 Australian Software Engineering Conference (ASWEC'04), IEEE Computer Society, 2004.

[11] S. Harker, K. Eason, and J. Dobson, "The Change and Evolution of Requirements as a Challenge to the Practice of Software Engineering", Proceedings of IEEE International Symposium on Requirements Engineering, (San Diego, CA, USA), pp. 266–272, 1993.

[12] J. Brier, L. Rapanotti, and J. Hall. 2006. Problem-Based Analysis of Organizational Change: A Real-World Example. In Proceedings of the International Workshop on Advances and Applications of Problem Frames.

[13] Mathisen E., Ellingsen K. And Fallmyr T. 2009. Using Business Process Modeling to Reduce the Effects of Requirement Changes in Software Projects. 2nd International Conference on Adaptive Science & Technology.

[14] A. Blyth, "Modeling the business process to derive organizational requirements for information technology," SIGOIS Bull., vol. 16(1), pp. 25–33, 1995.

[15] C. Ebert, "Understanding the Product Life Cycle: Four Key Requirements Engineering Techniques," IEEE Software, vol. 23(3), pp. 19–25, 2006.

[16] B. Nuseibeh and S. Easterbrook. 2000. Requirements Engineering: A Roadmap. In Proceedings of the Conference on the Future of Software Engineering.

[17] K. Orr, "Agile Requirements: Opportunity or Oxymoron?", IEEE Software, vol. 21(3), pp. 71–73, 2004.

[18] S. Harker and K. Eason. 1999. The use of Scenarios for Organizational Requirements Generation. In Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences.

[19] R. Kazman, G. Abowd, L. Bass, and P. Clements, "Scenario-based analysis of software architecture," IEEE Software, vol. 13(6), pp. 47–55, 1996.

[20] W. W. Royce, "Managing the development of large software systems." in Proc. Wescon., pp. 1–9, 1970.

[21] K. Beck, "Embracing change with extreme programming„" IEEE Computer., pp. 70–77, 1999.