Requirements Volatility in Software Development Process

M. P. Singh, Rajnish Vyas

Abstract- Changes in requirements do occur during the software development life cycle. The changes may take place from the initial design phase up to the implementation phase. These change that creep during the development process pose risk to cost and quality of the product, but at the same time provide an opportunity to add value.

This paper discusses the requirement, volatility in requirements, causes of requirement volatility and then the impact of requirement volatility on Project Schedule, Project Cost, Project Performance, Software Quality and Software Maintenance. We also try to explore the positive implications (if any) of the requirement changes. The purpose of this paper is to discuss aspects related to requirement volatility.

Keywords- Requirements change; requirements management; project management; card sorting; software evolution; development; maintenance.

I. INTRODUCTION

Requirements are the foundation of the software development process, as they provide the basis for estimating costs and schedules as well as developing design and testing specifications. So the success of any software project is directly related to the quality of its requirements. Although an initial set of requirements may be well documented, requirements will change throughout the software development lifecycle. This, constant change (addition, deletion and modification) in requirements during the development life cycle impacts the cost, schedule, and quality of the resulting product [4].

Ideally, the requirements once approved by the client should stabilize with no or very few major changes. According to *Capers Jones*, requirements change (RC) should come down to 3% in the design phase, 1% in the coding phase and ideally 0% during testing. However requirements change is always there but it can have very negative affect during the later stages of software development. For example: requirements change during the coding and testing stage can maximize the defect density as compared to other phase. Studies conducted by Jones have shown that the defect rates associated with the new features added during mid-development are about 50% greater than those of the artifacts associated with original requirements. [10].

There are many links in the requirements communication chain, as illustrated in Figures 1 and 2. A breakdown in any of these links leads to significant problems. For example, if an analyst misunderstands stakeholder input about requirements, if important requirements information does not surface or if an analyst and developer do not share the same understanding about requirements, the resulting product will not satisfy customers [2].

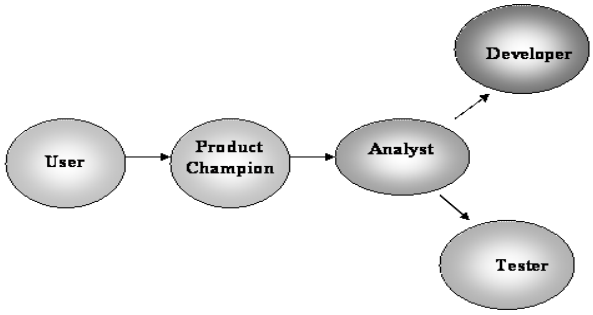

Figure 1: Requirements communication links in an information systems environment [2]

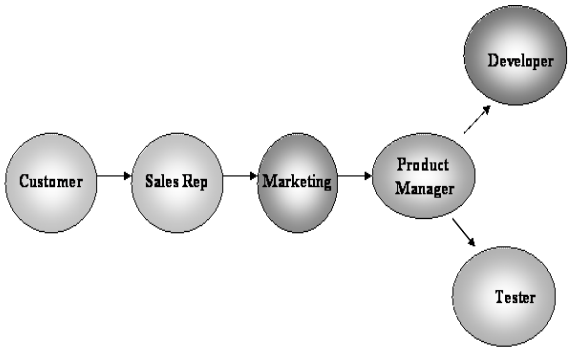

Figure 2: Requirements communication links in a commercial software environment [2]

The inevitable outcome of requirements errors is time consuming and costly rework, Analyst report that rework can consume 30 to 40 percent of the total effort expended on a software development project. Multiple studies have indicated that roughly 50% of the defects identified on software projects can be traced back to errors in the requirements. One analysis of the potential return on investment from better requirements suggests that requirements errors can consume between 70 to 85 percent of all project rework costs.

Figure 3 illustrates, it can cost up to 110 times more to correct the requirements defect found in operation than it would if the same defect had been discovered during the requirements definition [2].

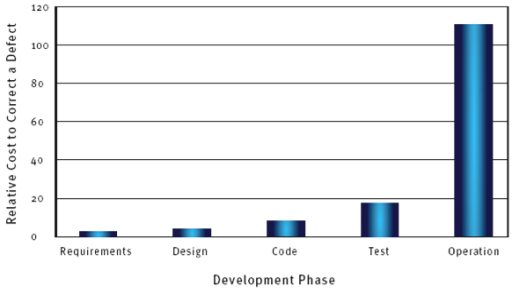

Figure 3: Requirements cost to correct a requirement defect depending on when it is discovered [2]

Manuscript received September 02, 2012.

Dr. M.P. Singh, Principal/Director, N.I.E.T, Alwar, Rajasthan, India
Rajnish Vyas, Research Scholar, Singhania University, Rajasthan, India.

This change in Requirements during the system development is known as Requirements Volatility. Requirements volatility often results in significant growth in requirements size from the time of initial requirements specification to final requirements of the system development. *Requirements volatility is an important risk in software project success that can occur in multiple points during the software development process.* These changes take place while the requirements are elicited, analyzed and validated and after the system has gone into service, simply through the software development lifecycle. These constant changes in requirements, known as Requirements Volatility, during the development lifecycle has great impact on the cost, the schedule and the quality of final product. Every phase of software development is effected by requirements volatility. *Many projects fail due to requirements volatility and some are completed partially [1]*. Though, Requirements volatility can not be overcome fully but can be minimized.

The requirements volatility has great impact for the success of software development. The chaos report says that the requirements volatility has 11% in total percentage causes in failures of software project [1]. Requirements volatility will affect the project schedule and cost overrun, project performance, and project quality. In fact every phase of software development is effected by requirements volatility.

Software development is considered to be a dynamic process where demands for changes seem to be inevitable [8]. Therefore the problem is not with requirements volatility, the problem is with inadequate approaches for dealing with them in a way that minimizes and communicates the impact to all stakeholders.

Our research work investigates both the pre and post-release requirements changes.

II. BACKGROUND

The Traditional Waterfall Approach

The activities that comprise the creation of software are commonly modeled as a software development lifecycle. The software development lifecycle begins with the identification of a requirement for software and ends with the formal verification of the developed software against that requirement. The software development lifecycle does not exist by itself, it is in fact part of an overall product lifecycle. Within the product lifecycle, software will undergo maintenance to correct errors and to comply with changes to requirements. One of the more generally accepted lifecycle models is the waterfall model (also known as the linear sequential model) as depicted in Figure 4.

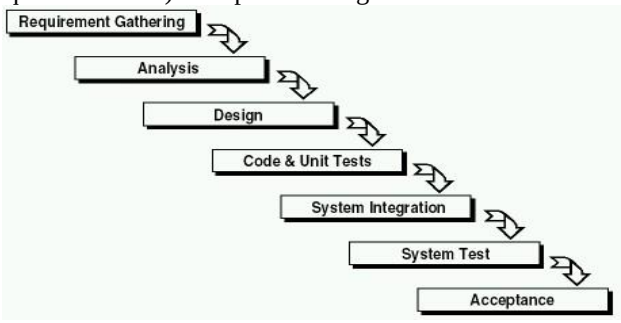

Figure 4 - The Waterfall Model

We see that the waterfall model is a sequential one, consisting of the following process areas:

- The requirements phase, in which the requirements for the software are gathered and analyzed, to produce a complete and unambiguous specification of what the software is required to do.
- The architectural design (analysis) phase, where software architecture for the implementation of the requirements is designed and specified, identifying the components within the software and the relationships between the components.
- The design phase, where the detailed implementation of each component is specified.
- The code and unit test phase, in which each component of the software is coded and tested to verify that it faithfully implements the detailed design.
- The system integration and system test phase, in which the software is integrated to the overall product and tested.
- The acceptance testing phase, where tests are applied and witnessed to validate that the software faithfully implements the specified requirements.

The Historical Perspective

The Waterfall Model outlines a sequence of activities that are still found in most of today's software development processes.

Leading the development process is the Requirements Analysis activity, followed by Design, Coding, and Integration & Testing. This conceptual "framework" provided significant insights as to how software should be developed. Furthermore, it paved the way for the definition of many similar models and paradigms that are composed of essentially the same basic set of activities (or phases. A closer examination of existing software development models reveals a common thread: they all contain a Requirements Analysis phase, it is usually one of the first phases in the model, and it is composed of a series of activities that relate to the gathering and analysis of requirements from the customer.

However lately, Software engineering community had given Requirement gathering activity lower priority than the development activity. Subsequently, only within the last few years have we seen a meaningful refinement of

"Requirements Engineering" that recognizes the major activities that underline requirements gathering.

According to Stark, total of 123 requirements changes were made to the 44 software releases under study. The releases that experienced requirements volatility, sixteen of the deliveries (36%) had no requirements change; of these, seven were made according to schedule, five more were within 15% of the original scheduled date, and five were more than 15% late. The remaining 28 releases (64%) were affected by requirements changes, with nine of them having greater than 50% volatility. *The average requirements volatility for these 44 releases is 48%.*

Taken literally, the data from these studies of multiple projects shows that at least one third and potentially as many as half of the delivered requirements implemented during system maintenance were not part of the original plan. *The largest observed volatility is an astonishing 600%.* Distribution of these *changes* by category may be done as, additions to the release functionality are the most common form of change (59%), followed by deletions (35%), with scope *changes (6%)* occurring least frequently.

III. DEFINITION OF REQUIREMENT

As seen above, every software development process starts with the requirements collection. The customer has some notion of what the system should do. We collect the customer needs for a software system and prepare the requirements documents according to the customer needs. Requirements document is a blue print of the system.

Requirements depend upon the project; they differ from project to project. Requirements can be of two types, functional and nonfunctional.

Functional requirements describe an interaction between the system and its environment. For example, calculating total marks of student in all subjects, giving the grade and rank.[1]

Nonfunctional requirements describe a restriction of the system that limits our choices for constructing a solution to the problem. For example hardware and man power resources etc [1].

Changes to requirements and change requests

Once we collect the requirements from the customer, we sign the agreement with the customer. In the agreement both the parties agree what are all the requirements to be developed in the system. These requirements are called baseline requirements.

Any changes to the baseline will be done only through the submission of change requests.

The following is a suggested process for requirements change management:

- The need for a change must be identified
- A change request is submitted.
- The change request must be approved or rejected.
- Approved changes are documented and incorporated into the baseline.
- Submission of a new baseline should involve a formal review.
- Submitted materials should contain sufficient details so that changes can be backed out if necessary.

All change requests will be reviewed on a regular basis by the project change control board. The change manager will drive the schedule based on the number and complexity of change requests and a Cost/Schedule Impact Analysis (CSIA) is demanded. The requirement change control team reviews the requested change and either accepts, reject, or defer and also ensure that the resources are neither scarce nor wasted during change activities.

IV. DEFINITION OF REQUIREMENTS VOLATILITY

There is no standard definition of requirements volatility. Usually it expresses the changing nature of requirements over the system development life cycle. There are many definitions of Requirements Volatility.

D Zowghi[3], describes requirements volatility as *the tendency of requirements to change over time in response to the evolving needs of customers, stake holders, organization, and work environment.*

We can take operational definition of Requirements volatility as the number of Requirements changes (addition, deletion, modification) to the total number of requirements for a given period of time [3].

The change to requirements—after the basic set of requirements has been agreed by both the clients and developers of the requirements – are known as requirements volatility [1].

Requirements changes will occur from the first phase (System/Information Engineering and modeling) to the final phase (Maintenance) of the software development life cycle, depending upon the time (phase) it occurs.

The changes can be described as pre and post release requirements changes.

It is important to understand that Requirement Change can be distinguished as:

1. Pre-release requirements changes:

a. Pre –FS (Functional Specification): Requirements changes that refer to changes to requirements during the early phases, elicitation, elaboration, analysis, modeling and negotiation of software development, before the functional specification has been completed and signed off.

b. Post –FS (Functional Specification): Requirements changes that occur during the later phases of software development lifecycle i.e. design, coding, development and testing, after the FS has been formally signed off by the client and product developers.

2. Post-release requirements changes:

The changes that occur once the system has been deployed at the client side, after the system has been released. This occurs in maintenance phase. In the above context, it is worth mentioning that the first type (1.a) of changes is constructive if correctly done, because these changes would help in defining more complete requirements. However the 1.b and second type of requirements can be destructive, as they affect the productivity in terms of cost overruns, schedule overruns and quality .Therefore, to keep track of Post-FS and post release changes to requirements is more important [1].

In the above context, *it is worth mentioning that the type of change under 1(a) is constructive if correctly done, because these would help in more complete requirements.* However the 1(b) and second types of requirements can be destructive as they may affect the productivity in terms of cost overruns, schedule overruns and quality *(adding defects while incorporating a change).*

V. CAUSES OF REQUIREMENTS VOLATILITY

Requirements evolution is due to Internal and External factors (social view point).

1. External Factors:
   a. Government regulations
   b. Market competitors
2. Internal factors
3. In technical view point
   a. Product constraints
   b. Lack of experience to the project development team professionals
   c. Feed back from other phases of SDLC
   d. Project size/ requirements overload
   e. Software (price changes) and hardware.
4. Potential for change (changes in business environment)
5. Requirements in stability (the extent of fluctuation in user requirements)
6. Requirements diversity (the extent to which stake holders disagree among

themselves deciding on requirements).

7. Requirements analyzability (the extent to which the process of producing requirement -specification can be reduced to objective procedure).
8. Poor communication between users and development team.
9. Client irresponsibility

Among the causes of requirements volatility, some can not be avoided; some other can be avoided by taking precautions. For instance, government regulations and market competitors can not be avoided. We can minimize the requirements volatility effects of communication between the customer and developer, client irresponsibility, feedback from other phases of software development life cycle.

From the sample metric report of requirement change release, we can say the dominant reason for requirement addition are changes in external interfaces and the second most frequent reason is oversight in an earlier version of specification. Customer request for enhancement will also play an important role in requirements volatility, while the others have a small impact [1].

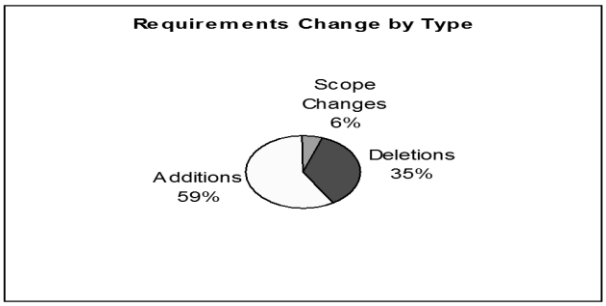

Figure 5: Requirements change by type [1]

The requirement changes, by type have been identified as additions (59%, deletions (35%) and modifications (6%) [1].

VI. HOW REQUIREMENTS VOLATILITY IS A RISK IN SOFTWARE DEVELOPMENT PROCESS?

Requirements volatility has a great impact on software development life cycle. The requirements volatility affects software releases and has major impact on project schedule, cost and project performance. The degree of requirements volatility is negatively associated with software project schedule and cost performance. Lamswede conducted a survey, from the data collected from over 8000 projects from 350 companies in USA and revealed that one third of the projects were never completed, and half succeed only partially, i.e. with partial functionalities. Many projects have cost overruns and significant delays [2].

We analyze the impact of requirement volatility in different phases of software development process, making it a major risk in different phases of development process.

6.1 Risk in Project Schedule

Software development life cycle includes many steps like, design, development and testing. However, before committing for software development, the customer usually wants to know how much the project will cost, the time it will take, and what are all the internal activities and milestones.

A project schedule describes the software development cycle for a particular project by enumerating the phases or stages of project and braking each into discrete tasks or activities to be done. The schedule also portrays the interactions among these activities and estimates the time each task or activity will take. Thus the schedule is a time line where the activities will have start and an end, and when the products will be ready [1]. The project schedule is the first task when we develop a software product. Software cost is also depending upon the project schedule. Software project schedule is consisting of intermediate deliverables, design documents, application source code, meeting with stakeholders to collect the requirements, and to approve the intermediate work.

When there are requirement's changes, these changes can affect design, code, and testing. The final effect of this requirements volatility can be missed deadlines. When you miss one deadline you need to postpone all subsequent deliverables deadlines. Sometimes you need to reschedule the total project schedule. A large number of software projects are stopped in the middle due to this reason.

Some times it is more important to develop a software project in the time scheduled rather than with high quality. For example, in banking sector when your competitor is introduced a new system offering the online banking facility you observed that your customers are moving to that bank attracted by new online banking facility and the new customers' rate is decreasing gradually. Therefore you decide to introduce online banking facility. You can not wait long time for this project. You need to implement it as soon as possible with minimum requirements facilities. Otherwise for each day of delay, you will loose old and new customers, so you need to complete the project with minimum high priority requirements.

6.2 Risk in Project performance

Requirements volatility has impact on the performance of the development lifecycle. As we discussed earlier it has impact on every phase of software development life cycle. The instability of requirements is characterized by the significant fluctuation of user's requirements in the later stages of the development. It is also characterized by the difference between requirements that were identified at the beginning of the project and requirements that existed at the end. The later aspect of requirements volatility is influenced by the differences and conflicts among users/stakeholders of requirements.

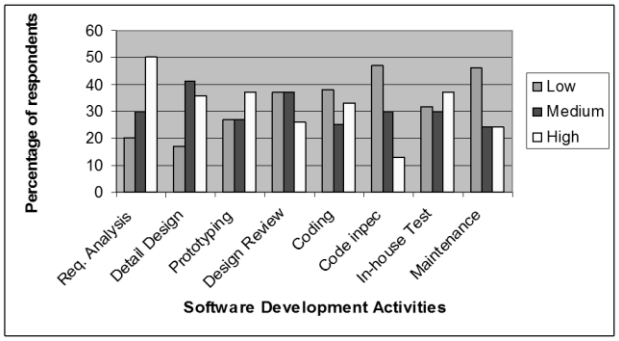

Figure 6: The level of requirements volatility from the respondent's perspective [3].

In a study performed by Zowghi and Nurmuliani, the developers were asked to rate the level of requirements change for each stage of software development activity. The result in figure 6 shows that most of respondents indicate that the level of changes during the requirements analysis is high. In the requirements analysis stage, software requirements are being explored, elaborated and fleshed out while the new

requirements are being discovered as the result of the analysis. At the end of prototyping stage, not surprisingly the requirements fluctuated as the users change their mind after they were shown early prototypes.

The level of requirements volatility seems to decrease as the project progress and near to the end of life cycle, but rise slightly again during coding and maintenance-house testing.

6.3 Risk in Project cost

Software project cost estimation is an important factor for software project managers. When we talk about cost effort it depends mainly upon project duration, number of employees you need(cost of employees), software and hardware efforts, building rents, traveling expenses and so on including requirements volatility. Based on the size of new requirements, we need to check which requirements will come under the baseline requirements and which are not.

For new requirements, developers need to make an agreement with new revised cost and schedule of the project.

Re-planning according to new requirements

When new requirements are received from the customers the project managers need to replan the schedule and the cost of the project. The developers need to negotiate with customers to make requirements tradeoffs, additions, and rejections. Developers could limit the functionality and defer the full functionality to the later releases. Re-planning of these new requirements is important because of their dependency on other requirements. Rework of software that implements the existing requirements will cost more, based on the new requirements.

Requirements volatility is important when calculating the project cost. The managers will quote the low price for the first stage of development and will fix higher price for the additional requirements changes (additions, deletions, modifications). Here you need to remember that the price of the product is decided by parties, developers and stakeholders with the initial requirements specification before the developers start the project.

6.4 Risk in Software maintenance phase

Large and long term applications will have frequent requirements changes. In maintenance environment, new requirements are added to release or existing requirements are deleted or modified. This will affect cost, schedule and quality of the software project. As we update the previous versions the quality of product increases. Stark said that in 44 releases the requirements volatility was 48% [4].

In maintenance phase the requirements volatility is caused mainly by government regulations, market competitors, and changes agreement between stakeholders, changes in business environment and changes in the regulations in their partner companies.

When in the release cycle do requirements changes occur?

An analysis of data shows that requirements volatility will affect the time in months of one project that was approved with 17 requirements and planned to be completed on a nine month schedule. The study shows that 20 total changes were made to the release content (volatility of 117%) in the 14 months since project plan approval. This change rate of 1.4 changes/ month (about 8% per month) is extremely high. [4]

6.5 Risk in Software Quality

There is no standard definition of quality. Software quality is said to be the customer satisfaction. If a software product meets all the requirements specification and it is bugs free then we can say that the software has high quality. However there is no bug free software till now. Software requirements volatility and software quality are inversely related to each other. As requirements volatility increases the reliability of the final product decreases. If the requirements volatility is low, the software quality is high.

The software is tested for bugs in the testing phase. Testing is performed done even in the coding phase, but that testing is only for that particular program, not directly related to all the system.

VII. IS VOLATILITY IS THE ONLY FACTORS THAT INFLUENCES THE SUCCESS OF A PROJECT?

IT projects can be disrupted by a variety of changes including technology, project requirements, personnel and the external environment.

Based on the various studies, we can see that requirement volatility is not the only reason contributing to the success of any project. There are many other factors that contribute to the success of a project.

According to Gulla 2011(Five Factor Model), various reasons contributing to the success of a project can be listed as below:

1. Project Management
    a. Plan
    b. Direct
    c. Solve Problems
    d. Communicate
2. People
    a. Skills
    b. Motivation
    c. Quantity
    d. Continuity
3. Business
    a. Alignment
    b. Funding
    c. Risk
    d. Return on Investment
    e. Data
4. Technical
    a. Hardware
    b. Software
    c. Testing
    d. Relationships between elements
5. Method
    a. Approach
    b. Procedures
    c. Tools

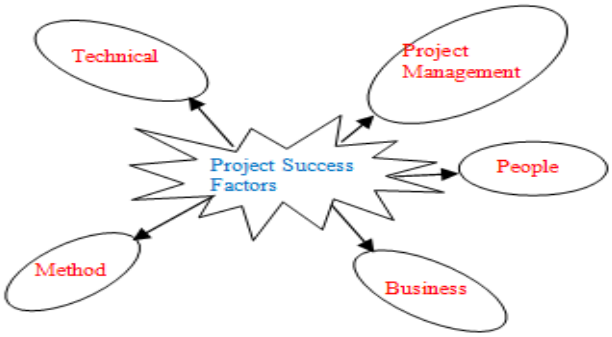

Figure 7: Factors influencing IT project failures

In 1994/95, the Standish group surveyed over 8000 software projects, and identified that the project failures is caused by poor requirements activities.

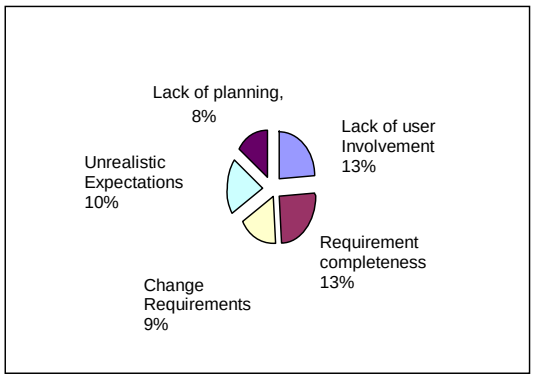

Figure 8: % of causes of Poor Requirement Activity responsible for Project Failure [12]

This data also supports that Change in requirement is not only the reason responsible for project failure, there are many other reasons contributing to the project failure.

VIII. CHALLENGES

During the study, it was observed that there are several challenges that lead to Requirement Volatility.

1. *Change request form was not always complete.*

It has been observed that the change request form, which is the sole tool to raise a change to the project deliverables, had little information about reason for the proposed change.

2. *No formal impact analysis and incomplete change effort estimation.*

The change request forms are generally incomplete, and the information available was not adequate to analyze the impact of the change. There was no formal impact analysis performed and the developer was frequently unable to predict the impact of potential change on some areas.

3. *Traceability between requirements and other software artifacts was not established.*

Requirements and the system design are linked; also requirements and the underlying reasons why these changes were proposed are also linked. When the changes are proposed, we have to trace the impact of these changes on other requirements and the system design. If traceability is not established, it is difficult to find the actual impact of the proposed changes.

IX. CONCLUSIONS

In this paper we have discussed several aspects of requirements volatility, in particular the causes and impact of requirement volatility on software development process. The causes of requirements volatility can not be overcome fully but we can minimize some causes like technical aspects and poor communication between stake holders and developers.

Requirements volatility has impact on the whole software development life cycle. It mainly affects the coding and maintenance phases of large and long term projects.

The impact of requirement volatility on the software project has been seen as:

1. Projects Schedule: If the schedule of one activity delays, obviously all subsequent activities schedule will be disturbed. Sometimes we need to reschedule the whole project.
2. Project Performance: Project performance decreases due to change in requirements. Requirement Volatility has high impact on coding and maintenance phases.
3. Project Cost: Project cost increases due to change in requirements.
4. S/W Maintenance: In this phase, requirement change is mainly due to Govt. regulations and market competitors. We need to revamp according to new requirements.
5. S/W Quality: Quality of software decreases due to continuous change in requirements. We need to RE design the test cases according to new requirements.

Though, requirements volatility has impact on project schedule, project performance, project cost, software maintenance and S/W quality, but it may have some positive effects as well as it may help us to have better understanding of user requirements.

Due to the impact of requirements volatility many projects have failed. Thus we can conclude that requirement volatility is a major risk in software development process.

FUTURE DIRECTIONS

From the management perspective, we wish to look at the impact of the requirements volatility in various phases of software development process, and after having observed that it is a major risk in software development process we will try to identify its positive impact also.

REFERENCES

1. Mundlamuri Sudhakar, “Managing the Impact of Requirements Volatility”, Master Thesis, 2005, Department of Computing Science, Umeå University, SE-90187 Umeå, Sweden.
2. “Effective Requirements Definition and Management”, April 2006, http://www.borland.com/resources/en/pdf/solutions/rdm_whitepaper. pdf
3. D. Zowghi, “A Longitudinal Study of Requirements Volatility in Software Development”, in the ASMA/SQA Meeting, 2005.
4. Zowghi, N. Nurmuliani, “A study of the Impact of requirements volatility on Software Project Performance”, Proceedings of the Ninth Asia-Pacific Software Engineering Conference , APSEC 2002, Gold Cost, Queensland, Australia,04-06 Dec 2002, pp:3-11.
5. Donald Firesmith: “Prioritizing Requirements”, in Journal of Object Technology, vol. 3, no. 8, September-October 2004, pp. 35-47.
6. http://www.jot.fm/issues/issue_2004_09/column4
7. Lamswede, A. Requirements Engineering in the Year 00: A research perspective. In proceeding of the 22nd International conference on Software Engineering (ICSE’2000), Limerick, Ireland, 5-19, ACM Press.
8. D.Zowghi, Susan. P. Williams, “Requirements Volatility and its Impact on Change Effort: evidence-based Research in software Development Projects”, in AWRE2006.
9. Ian Soumerville, Software Engineering, 7th Edition, Pearson Education
10. Pfleeger, S.L . Software Engineering Theory and Practice, Prentice Hall, 1998.
11. Jones, C. (ed) (1997): Software Quality: Analysis and guidelines for Success, International Thomson Computer Press.
12. George Stark, et al. “An Examination of the Effects of requirements Changes on Software Maintenance Releases”, Journal of Software Maintenance
13. Standish Group Report (CHAOS), 1995.