Causes and Impact of Slipped/Misunderstood Requirements on Large Scale Software Projects

Muhammad Adeel Ashraf, MehwishAslam, Shahid M. Awan
Department of Computer Science, School of Systems and Technology,
University of Management and Technology
Lahore, Pakistan
Email: adeelashrafcs@gmail.com

Abstract— **Requirement engineering is the first and foremost phase of a software development life cycle. Making mistakes during this process has enormous negative impact on all underlying activities of software development. Defects discovered when a system is deployed cost fifty to two hundred times more than defects discovered during software elicitation phase. In this research paper causes of missed requirements have been identified by literature review and supported by interviews and surveys from leading software industry professionals. In total, sixteen factors of missed requirements have been identified and divided into three categories namely; User factors, Analyst factors, and Common Factors. Understanding of these factors will aid in developing softwares with complete requirements and thus achieving customer satisfaction level and to avoid wastage of effort and poor quality software.**

Keywords: missed requirements factors; survey report; interviews; statistical analysis

I. INTRODUCTION

The communication initially starts with customers/stakeholders for requirements gathering and this process continues for the entire time during project development, involving all the stakeholders. Communication continues between all the project team (Requirement engineers, software developers and tester). To make a software project successful, beside the need of different tools and techniques, effective communication plays a vital role. Errors made during requirement gathering phase can be responsible for 60% of the cost of the project, delayed schedules and ultimately resulting in customer rejection for software developed.

Past studies of requirement engineering emphasizes on DIFFERENT REQUIREMENT ELICITATION WAYS WHICH ARE MODELS, TOOLS and functions to identify the causes of weak requirements. Purpose of this paper is to identify causes of poor requirements and effects of these requirements on the project quality.

Software with incomplete requirements is always undesirable by customer. Efficient requirement engineering plays a vital role. Understanding the impacts of missed requirements due to poor requirement engineering is essential to minimize the risk of missed requirements. Poor quality requirements result in failure to meet customer expectations, poor quality software and wastage of time and efforts. A high understanding of all the possible factors affecting requirement gathering is important to deliver complete, desired and efficient software. We can maximize the software efficiency by closing all the requirement elicitation holes and thus delivering software that fulfills all the needs and expectations of customer.

Misunderstood requirements by developers and users always result in 'Misunderstood or skipped requirements'. Finding these causes of missed requirements and avoiding these during requirement elicitation phase can improve the quality of software.

The main purpose of this study was to find realistic ways to bridge the gap between user and developer to create improved requirement elicitation process. By doing so customer satisfaction level could be increased and project cost could be decreased [9].

II. RELATED WORK

Major studies to understand software development life cycle suggest that the proper communication between customers, development engineers and requirement engineers is an essential part of correct understanding of the requirements [3]. But there is a problem in large organizations because large organizations hide some business critical information which is essentially important for software requirements. It was also noted that the proper communication cannot be fulfilled through documentation. Interaction and communication with different people plays an important role in requirement engineering.

Designing of large software system problem were studied by conducting interview through a team assigning different projects almost seventeen. For this purpose a model is used to find and analyze the problem which is called behavioral model. Different type of flaws were observed like knowledge about domain is very thin and communication gaps between customers, development engineers and requirement engineers. He also observed the pros and cons on software developments and quality through these gaps. This activity is observed conducting an interview of different positions at managerial level [3].

Communication has also been reported challenging for distributed software projects. In this field the efforts of Holmstrom et al, Kotlarsky and Oshri, Piri are significant [4]. Holmstrom et al. point out temporal space as tough in everyday communication in universal software development environment. Additionally, even in general software development projects where agile methods were used communication have also been noted as challenging. On the other hand, Kotlarsky and Oshri reported that challenges

included in sharing knowledge across internationally distributed teams are still common [4]. Finally, Piri reports that many of the common problems meet in software development projects can be drawback to social factors of the project with special challenges to communicate among dispersed teams [5].

Al-Ani and Edwards worked on communication models [1]. Lutz worked on linguistic challenges [6]. Niinimaki et al's report on finding communication tools in twelve distributed software projects is mentionable [7]. The relations between individuals with different people in cross-functional developing teams have been calculated and the many of absent communication boundaries was found between people performing different roles that were not supposed to be communicating according to the formal organizational structure.

III. Research Methodology

Quantitative research approach has been used for finding and verification of missed requirements causes, which is suitable when individual thinking of a complex problem is to be studied, using interview methodology. The research has been carried out in three stages, described as follows.

A. Literature Review and Hypothesis Generation

To find the causes of missed requirements we chose to study the literature provided by authors [1] which is used as input in identifying the 'assumed' factors involved in poor requirement elicitation process and hypothesis generation. To avoid biased thinking in selecting causes of missed requirements described by only one author, the process of selecting assumed factors have been justified by extensive study of literature review provided by other authors as well and brainstorming sessions with hypothesis by authors this paper. Outcome of literature review has been used as input for conducting interviews with analysts of software elicitation process and designing the questionnaire, which is used to validate the factors selected for missed requirements causes by literature review and interviews [8][10][12][13][14].

Following assumed factors of missed requirements have been identified by literature review activities which are further divided into three categories. User factors [UF], Analyst factors [AF], Common factors between user and analyst [CF].

A.1 User Factors Responsible for Missed Requirements [UF]

Following factors have been identified as cause of poor requirement elicitation process contributed by User.

A.1.1 User with incomplete understanding of his needs [UF1]

Most of the times customers are not clear about what exactly they want and thus unable to convey what exactly to built. This also happens if requirements are gathered from high level management of organization who can only give high level abstract information of the system.

A.1.2 Poor/Over Customer Involvement [UF2]

Sufficient customer involvement during software requirement elicitation process is very necessary as the customer is important element to provide requirements to analysts. Poor customer involvement in elicitation process leads to unclear and unrealistic requirements Most of the time users believe that analysts already know what is required by users and thus unintentionally hide major aspects of requirements which contribute to missed requirements and development of a product which was not desired by user. Over involvement of user into software elicitation process makes this process tiresome and confusing for analysts too.

A.1.3 Conflicting views of stakeholders [UF3]

Different stakeholders have different priorities for the system to be developed and thus they convey requirements according to these priorities. It also involves political environment of an organization.

A.2 Analyst Factors Responsible for Missed Requirement [AF]

Following factors have been identified as cause of poor requirement elicitation process contributed by requirement engineer/analyst.

A.2.1 Missed Requirements [AF1]

Missed requirements are disastrous for a software project. Software with complete needs of user cannot be developed with missed requirements. Identifying requirements later in product lifecycle causes wastage of resources. Complete requirements support the developer of software which is up to the mark of customer expectations.

A.2.2 Poor Quality Requirements (incomplete, inconsistent, inaccurate) [AF2]

Poor quality requirements involve incomplete, inconsistent or inaccurate requirements. A requirement taken with these attributes is of no use. Requirements should be complete, consistent and accurate to be considered as quality requirements [10].

A.2.3 Fuzzy and Ambiguous Requirements [AF3]

Fuzzy or ambiguous requirement are unhealthy for software. A product is facing ambiguity problem if a requirement mentioned in a document have several meanings and different readers interpret a requirement in different ways [8].

A.2.4 Uncertainty over Requirements [AF4]

Uncertainty problem over requirements occur when expectations level of different stakeholders (executives, developers) from the product is different. Secondly this problem occurs when requirements are given by users but analysts are really not sure about what to develop. Normally it happens when requirement discussions are mainly focusing functionality part of the product and as a result some expectations of the stakeholders are left unmentioned.

A.2.5 Unprioritized Requirements [AF5]

Requirements are to be prioritized according to their importance for the system. Prioritizing requirements helps product managers in activities like staff allocation, scheduling of resources and trade off between requirements [8][10].

A.2.6 Untrained Analyst [AF6]

Requirements can't be fully and efficiently gathered if requirement engineer is not fully trained. A requirement

engineer is responsible for capturing a big picture, understand it and describe it. He is also responsible for communicating with non-technical people as well as technical people which requires a training regarding software requirement engineering.

A.2.7 Unnecessary Requirements/Gold Platted Requirements/Scope Problem [AF7]

Scope sneak mostly happens when the product scope is not clearly defined. If new requirements sneaks in or sneak out, product scope definition becomes questionable. At times unnecessary gold platting to requirements is done by analysts causing wastage of time and resources and indulging with unnecessary requirements [14].

A.2.8 Poor Change Process Planning and Effect Analysis [AF8]

A project must have a defined process for dealing change in requirements otherwise a new functionality will be shown only in testing phase which is too late. Developers might implement changes which are already rejected or implement changes which are yet to be approved. The authority for change control must be well defined and changes must be communicated to all people getting affected. Effect of the change must be analyzed. The change can make the product more complex, effect schedule, technically infeasible or it can make a product over budgeted [8].

A.2.9 Version Control of SRS [AF9]

Version control of SRS plays an important role in efficient software development. A developer can implement a functionality which was removed from new SRS because the developer was not given an updated version of SRS. Testers can test the software against removed functionality of the product [8].

A.2.10 Inadequate Requirement Tool Support [AF10]

Adequate tool support is not used by many requirement engineers. Many times it happens that only SRS is used as requirement repository. Few analysts use requirement management tools like Borland CAliberRM, Telelogic's DOORS. Diagrams are major part of requirement engineering and this diagram gives a clear picture of requirements along with their attributes and develops a clear understanding throughout the project life [8].

A.3 Common Factors (user and analyst) Responsible for Missed Requirements [CF]

Following factors have been identified as cause of poor requirement elicitation process contributed by both user and analyst.

A.3.1 Language Barrier between User and Analyst [CF1]

Sometimes it happens that requirement engineering team and stakeholders are from different backgrounds and speak different language or variation in same language and thus unable to understand each other's product expectations fully.

A.3.2 tiptoe requirements/changing requirements [CF2]

A critical problem in requirement engineering occurs when requirements keep on changing even after requirements are finalized and development is started.

A.3.3 Inadequate Requirement Validation by Stakeholders [CF3]

A major task of analyst is to get the requirements validated by stakeholders in order to a get the requirements clearly specified and agreed by stakeholders. This problem can be occurred in two ways. One way is if stakeholders are not accessible and analysts don't have adequate access to stakeholders. Second way is if software requirement team doesn't perform all the requirement tasks and thus eliminating requirement validation by stakeholders which may be due to product schedule or project budget [8].

B. Interview with 12 Software industry persons

Semi structured Interviews were conducted with 12 experienced software engineers from software industry. Hypothesis, based on 16 assumed factors for missed software requirements was discussed with them and requested to add any other factor that they feel is responsible for missed requirements with the help of their software industry experience but not a single additional factor could be generated by interviews.

B. Survey by questionnaire

Hypothesis of 16 assumed factors of missed requirements was presented 40 software industry persons via questionnaire [appendix A] to get the degree of agreement with the hypothesis. The degrees like experienced, strongly agree, partly agree, disagree were presented against each of 16 assumed factors and their opinion was gathered and statistically analyzed [11].

IV. SURVEY RESULTS

Following results have been found from survey of 40 respondents of software industry. The percentage of degree of agreement against each factor is shown in the Table I.

TABLE I. SURVEY RESULTS

Survey Results	Experienced	Agreed	Partially Agreed
1:User with incomplete understanding of his needs	80%	20%	0%
2:Poor/Over involvement of user	50%	50%	0%
3:Conflicting views of stakeholders	40%	50%	10%
4:Missed requirements by requirement engineer(analyst)	30%	60%	10%
5:Poor quality requirement(incomplete, inconsistent)	50%	50%	0%
6:Fuzzy and ambiguous requirements	40%	60%	0%
7:uncertainity over requirements	50%	50%	0%
8:unprioritized requirements	30%	60%	10%
9:untrained analyst	20%	50%	30%
10:Scope problem/gold platted requirements	60%	40%	0%

11:poor change control of requirements	20%	60%	20%
12: version control of SRS	20%	40%	40%
13: inadequate requirement tool support	30%	50%	20%
14:Language barrier between user and analyst	30%	40%	30%
15:changing requirements	70%	20%	10%
16:inadequate requirement validation by stakeholders	30%	50%	20%

This form is also online and the link is given in link [11].

Top three causes of missed software requirements analyzed by survey result are as follows: User with incomplete understanding of his need, Changing requirements, and Scope problems.

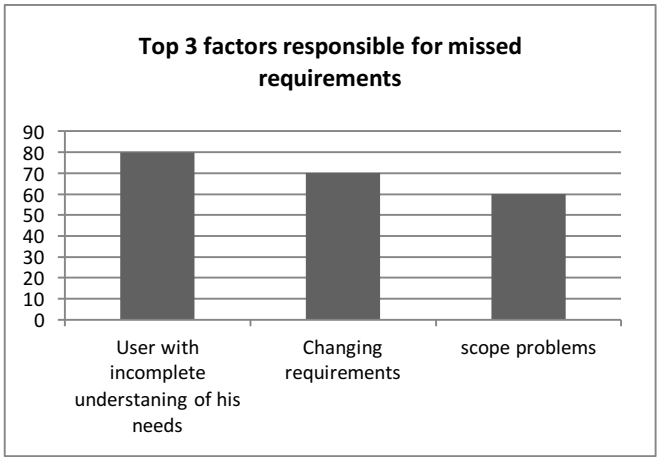

Fig. 1. Percentage of top three factors analyzed by survey results.

V. Analysis Of Result And Recommendations

Recommendations are for each factor as numbered in Table 1.

R1: Most of the time the customer assumes that he/she has fully described the system and has given all the inputs required for the system to be developed but in reality this is not the case. Analyst should gather requirements of large software systems not only by interviewing the customer but also some other methods of requirements elicitation should be used. Ideally the requirements should be gathered by end users of the system although the vision of organization perceived by top executives of organization should be considered too.

R2: Requirement elicitation is a collaborative process between user and analyst. Active participation of user is necessary in this process to get the maximum output of requirement gathering activity. Over involvement of user should also be discouraged because it can make this whole process confused and tiresome for analysts.

R3: Finding out the real stakeholder of the system is a difficult but important task. Project started by one stake holder can be cancelled by another so examining the real stakeholder who has the final authority regarding the project is really essential. Internal conflicts should not have impact on requirement elicitation process.

R4: Missed requirements are much difficult to point out as compared to poorly specified requirements. Missed requirement problem usually cannot be spotted until the system is developed or deployed. Analyst must gather requirement carefully in spite of fully depend on stakeholders or users. Elicitation process must involve all possible stakeholders and requirement engineering team members so that this process to be carried out efficiently. Methods and techniques like use case modeling and swim lane diagrams must be made with all preconditions and post conditions.

R5: Gathering quality requirements is a difficult task and for this purpose not only analysts but reviewers and inspectors should be trained. They should be able to differentiate between good and poor requirements. Reviews to ensure quality requirements are necessary. Engage members of design and test teams to ensure feasible and verifiable requirements.

R6: Analysts should review and inspect all requirements to verify and ensure that none of the requirements is fuzzy or ambiguous. Each requirement should be checked against a checklist of most common ambiguity defects. Tools should be used to filter out any vague words used in requirement document.

R7: Business, user and functional requirement should be dealt separately. All of them should be considered important to eliminate the uncertainty factor.

R8: setting a priority level for each functional requirement is an important attribute. Implementation of truly essential functionality is carried out by deriving functional requirements from use case description. Allocating each requirement to a particular build is the key.

R9: Characteristics of good requirement engineer are trained, experienced, motivated, communication skill and domain knowledge. Analysts should be properly trained to gather good quality requirements which can be done by classes, workshops, tutorial, books and by giving them chance to work with more experienced people.

R10: A full stop can never be marked against requirements as requirements keep on adding and enlarging the scope of the project. Use of modern lifecycle to allow addition of requirements can be a good choice to handle scope problems.

R11: Change control process must be defined for your project. A change control process must be supplemented change tracking tool. However, remember that a tool is not a substitute for a process. Set up a change control process to consider proposed changes at regular intervals and make decisions to accept or ignore them.

R12: If versions of SRS are not fully updated it may cause development of invalid requirements. Each change should be updated in SRS and latest version of SRS should be distributed to developers and testers.

R13: A powerful requirement management tool for storing metadata of requirements is very important. It enables analyst to keep record of requirements with the help of diagrams to have a clear picture of requirements. With the help of tools forward and reverse requirement engineering can be achieved. Traceability is done easily with these tools.

R14: Language should not be a barrier between analyst and user. The analyst must be of same lingual background to

efficiently understand customer needs.

R15: The idea of freezing the requirement at specific milestones is worth trying but frozen requirements should be placed under configuration control and all the impact of change needs to be determined before the changes are decided to take place. A close eye should be on budgets and schedules incase of changing requirements.

R16: Analyst should make sure that requirements are validated by stakeholders to ensure that all requirements are correctly specified and up to their demands. Requirement validation by stakeholders should be a functional part of project's schedule and budget.

VI. Conclusion

Quality software path begins with good requirement engineering process. For requirement to be of good quality, an efficient requirement engineering process is needed. An efficient requirement elicitation process can be achieved by educating all stakeholders including user's manager's requirement team members about requirement engineering process and application domain. There should be collaboration between customer and developer for requirement elicitation and management. All the requirements should be classified and divided in appropriate categories. An iterative and incremental approach should be used for requirement engineering process and standard template should be used for SRS. Vision and scope of organization should be kept in mind while developing software. Formal and in formal reviews of requirement document is very essential. Test cases for requirements and prioritization of requirements is necessary. All the changes must be incorporated efficiently. Keeping all these guidelines can ensure good quality software.

VII. Future Research Directions

Future work includes investigating additional factors causing missed software requirements and their impacts on software requirements elicitation process and quality of the product to be developed and relationship between these factors.

Acknowledgment

We would like to thank all our interviewees and questionnaire respondents from software industry who contributed to the fulfillment of this research project.

References

[1] B. Al-Ani, H. K. Edwards, "A Comparative Empirical Study of Communication in Distributed and Collocated Development Teams." Proc. IEEE Int. Conference on Global Software Engineering (ICGSE '08). IEEE Computer Society, pp. 35-44. doi: 10.1109/ICGSE.2008.9.

[2] A. J. Coffey and P. A. Atkinson, "Making Sense of Qualitative Data: Complementary Research Strategies", Sage Publications, Inc, 1996.

[3] B. Curtis, H. Krasner, and N. Iscoe, "A field study of the software design process for large systems." Commun. ACM, vol. Nov. 1988, pp. 1268-1287, doi:10.1145/50087.50089.

[4] J. Kotlarsky and I. Oshri. "Social ties, knowledge sharing and successful collaboration in globally distributed system development projects." Eur Journal of Inf Systems, vol. 14, Mar. 2005, pp. 37-48, doi: 10.1057/palgrave.ejis.3000520.

[5] A. Piri, "Challenges of Globally Distributed Software Development - Analysis of Problems Related to Social Processes and Group Relations", Proc IEEE Int Conf on Global Software Engineering, Sep. 2008,pp. 264-268, doi: 10.1109/ICGSE.2008.33.

[6] B. Lutz, "Linguistic Challenges in Global Software Development: Lessons Learned in an Int. SW Development Division,", Proc. Int. Conf. on Global Software Engineering, Jul. 2009, pp. 249-253, doi: 10.1109/ICGSE.2009.33

[7] T. Niinimaki, A. Piri, C. Lassenius and M. Paasivaara, "Reflecting the Choice and Usage of Communication Tools in GSD Projects with Media Synchronicity Theory" Proc. IEEE Int. Conf. on Global Software Engineering, Sep. 2010, pp. 312, doi: 10.1109/ICGSE.2010.11

[8] Karl Wiegers and Joy Beatty. "Software Requirements Engineering", 2nd Edition.

[9] E. Bjarnason, K. Wnuk and B. Regnell, "Requirements are slipping through the gaps — A case study on causes & effects of communication gaps in large-scale software development," 2011 IEEE 19th International Requirements Engineering Conference, Trento, 2011, pp. 37-46.

[10] Firesmith, Donald. "Prioritizing Requirements." Journal of Object Technology 3.8 (2004): 35-48.

[11] https://docs.google.com/a/umt.edu.pk/forms/d/1JYA0U73hx8ZASOfsZ0-5PInVSfSz609xyskx_1MP-9c

[12] Pohl, Klaus "Requirements EngineeringFundamentals, Principles, and Techniques" 3rd Edition.

[13] ElizebethBjarnason"Challenges and Practices in aligning requirements with verification and validation".

[14] ElizebethBjarnason"Causes and effects of over scoping in large scale software projects".