Analysis of Requirements Volatility during Software Development Life Cycle

N Nurmuliani, Didar Zowghi, Sue Fowell
Faculty of Information Technology
University of Technology, Sydney
P O Box 123 Broadway
NSW 2007, Australia
{nur, didar, sfowell}@it.uts.edu.au

Abstract

Investigating the factors that drive requirements change is an important prerequisite for understanding the nature of requirements volatility. This increased understanding will improve the process of requirements change management.

This paper mainly focuses on change analysis to identify and characterize the causes of requirements volatility. We apply a causal analysis method on change request data to develop a taxonomy of change. This taxonomy allows us to identify and trace the problems, reasons and sources of changes. Adopting an industrial case study approach, our findings reveal that the main causes of requirements volatility were changes in customer needs (or market demands), developers' increased understanding of the products, and changes in the organization policy. During the development process, we also examined the extent of requirements volatility and discovered that the rate of volatility was high at the time of requirements specification completion and while functional specification reviews were conducted.

Keywords: requirements volatility, taxonomy of change, causal analysis

1. Introduction

Despite advances in Software Engineering over the past 30 years, most large and complex software projects still experience numerous changes during their life cycle. These changes are inevitable and driven by several factors including constant changes in software and system requirements, business goals, market demand, work environment and government regulation [1].

Software development is a dynamic process. This often causes software requirements to change while development is still in progress. If these changes to software requirements are frequent then they may produce significant project uncertainty. Requirements change has been reported as one of the main factors that cause a project to be challenged [2, 3]. This indicates that managing requirements change still remains a challenging problem in software development.

Although the intention is for software requirements specifications to be captured and formed correctly in the initial stage of development, requirements inevitably change throughout system development and maintenance process. As a consequence, we need to identify a better approach to manage the impacts of continuously changing requirements. We believe that identifying and understanding the underlying causes of requirements change is the first step towards better and effective management requirements change in this rapidly changing environment

In this paper we present a qualitative method to characterize and evaluate requirements change problems throughout system development process. We apply this method to analyse requirements change management process in a large multi-site software development company. This leads us to develop a taxonomy that can be used as an approach to classify requirements change and to identify the causes of these changes. The results improve our knowledge and understanding of requirements volatility. This increased understanding will improve the process of requirements change management.

Proceedings of the 2004 Australian Software Engineering Conference (ASWEC'04)
1530-0803/04 $ 20.00 © 2004 IEEE

In the next section we present the background of requirements volatility study. We then briefly describe the organisation where this case study was conducted in section 3. In section 4 we present our data analysis framework and illustrate the procedures for conducting causal analysis of requirements volatility. We present the details of our findings in section 5 and we conclude this paper with discussion and future work.

2. Background

Requirements volatility (RV) is generally considered as an undesirable property. It has the potential to produce adverse impacts on the software development process [4]. Previous studies have identified that requirements volatility causes major difficulties during development. For example, a field study conducted by Curtis et al [2] indicates that requirements volatility is one of the major problems faced by most organisations in the software industry. Boehm and Papaccio [5] have observed that requirements volatility is an important and neglected factor that can cause software cost overrun. Other requirements volatility problems have been identified such as unstable or changing requirements during elicitation process [6] and during maintenance process [7].

Requirements volatility is also a common phenomenon that is present in most software development projects. However, very little research has been published on the identification of requirements volatility problems and the strategies to manage its impact on software development projects. Recent empirical studies have investigated the impact of requirements volatility on the software project schedule during maintenance [7], on software defect density during code and testing phases [8], on development effort [9] and on software project performance [10]. These studies indicate that requirements volatility is an important issue in system development and maintenance process.

While the existence of requirements volatility cannot be ignored, there is still a need to improve our understanding of requirements volatility problems in order to better manage its impacts. The first step to achieving this goal is to characterize and evaluate the problems of requirements change (i.e. reasons and sources of changes). Only then the causes of requirements volatility can be identified.

Harker and Eason [11] suggested that classifying requirements changes is one of the important factors that need to be considered in managing requirements change. They distinguished between stable and volatile requirements (emergent, consequential, mutable, adaptive, and migration). Classifying changes will help software developers to analyse each type of change according to its origin and assess its impact on software development process and product. Other studies [12, 13] also suggested that a classification of change is a scientific step to improve our ability in understanding requirements evolution.

Few studies have discussed and highlighted issues that relate to the causes of requirements volatility. Christel and Kang [6] indicate that requirements volatility is one of the main problems during the requirements elicitation process. The problem is triggered by continuous change in users' needs, disagreement among customers or stakeholders on agreed requirements, and changes in organization goals and policies. Other studies have mentioned contributing factors to requirements change, such as developers' knowledge of the application and business, competitors' products changes in technologies, poor communication between users, customers, stakeholders, and developers contributing to requirements change during system development [2, 14].

There is limited empirical research about requirements volatility. The concept is still not well defined in the literature. In this study we define requirements volatility as: *the tendency of requirements to change over time in response to the evolving needs of customers, stakeholders, organisation, and work environment.* The operational definition of requirements volatility can be represented as: *the ratio of requirements change (addition, deletion, and modification) to the total number of requirements for a given period of time.*

3. Case Study Context

The case study was carried out at Global Development Systems (GDS)[1]. GDS is an ISO 9001 certified software development company that belongs to an international multi-site organization with headquarters and marketing divisions in USA. It is an engineering lab that develops product line software. The software produced is characterized by the delivery of a series of releases. Each release is around 8000KLOC, development time of between 12-18 months, with approximately 180 full time developers involved. The product is an enterprise software, of which customers are themselves developers using the system for developing software. Requirements for new releases are requests for enhancements to the product and they are gathered from multiple sources:

[1] The company and product names are fictitious to preserve confidentiality.

- Market needs (representing current customers needs and market directions representing potential for future customers)
- Product strategy requirements (representing technology and engineering direction of the product in line with the organizational strategy)

At GDS, key stakeholder groups are scattered across several continents. The product strategy is directed from the US, where the Product and Program Management group is located across four sites. The development group is located in three Australian and one New Zealand sites, and customers are grouped in five large market segments across five continents. In addressing the geographical distribution of customers worldwide, the organization maintains on-site field support centres, to provide services to the diverse market segments.

The purpose of this paper is to identify and characterize requirements volatility problems and its underlying causes during the system development life cycle. Our unit of analysis to achieve this study objective is based on a single project. This paper presents our findings from one of the software releases. A waterfall model is applied to develop this release. During system development, all changes to products are documented and recorded in the project database. These activities enable us to inspect the documents and conduct an empirical analysis.

4. Research Approach

A single case study design with a single unit analysis (i.e. one project) in an industrial setting (GDS) was applied to investigate the problem of requirements volatility during software development. This approach is appropriate for the researcher to conduct in-depth investigation the situation of a typical project in the real software industry environment [15]

A historical change database was used to analyse the nature of requirements change. Change requests data were collected from one project release within GDS organisation. A total 0f 78 change request was collected during the last six months of the project duration (16 months).

4.1. Analysis Method

The purpose of our analysis was to identify and understand the problems relating to changing requirements during the software development process and their underlying causes. Our analysis is based on descriptive and qualitative methods.

Descriptive analysis provides rich information for understanding the requirements volatility problems as well as related aspects such as organizational policy, customer needs and product changes. Qualitative methods are employed to analyse the collected data and to evaluate the change process.

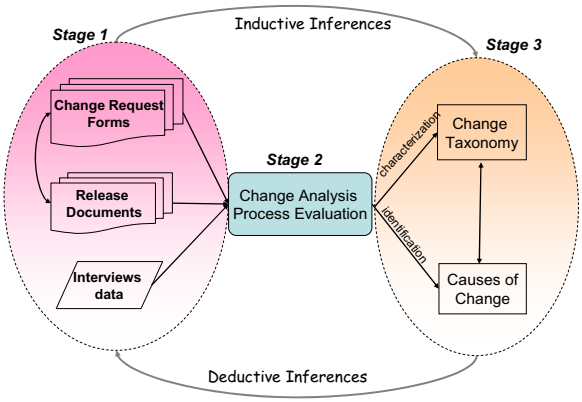


Figure 1. Data analysis framework

The data analysis framework we present in this paper is adapted from the general approach of Briand et al [16]. This method is used to determine the causes of requirements volatility and its related aspects. Figure 1 illustrates our data analysis process, which is a combination of both inductive and deductive inferences. Our approach will be described in the next section.

4.2. Causal Analysis of Requirements Volatility

The analysis process began by collecting change request data. The change request documents were collected, screened, and analysed. This paper focuses specifically on the analysis of change requests that related to requirements change. Our data analysis framework (as described in Figure 1) comprises three stages:

Stage 1: Understanding the changes

We considered three main sources of evidence to perform causal analysis of requirements volatility: Change Request (CR) forms, other release documents (i.e. requirements specification document, the configuration management plan, and software product documents), and interview data.

Based on the information contained in the change request forms, we identified problems related to each change. These include: description of the change, reasons for changes (*why factor*), types of change (addition, deletion, and modification), impacts of

change on software products or documents (*what factor*), effort estimate, elapsed time, and the person who requests the change. This stage represents the main part of the causal analysis process to characterize the causes of requirements change.

In the inspection of the change request forms, we often needed to crosscheck the content of the change form with other related release documents, such as requirements specification, requirements database, and software product documents. The purpose of this activity is to confirm our evaluation and triangulate our findings.

The other source of the change analysis process involves interviews with key figures in the project. Our aim being to capture information that was not available in the change request form. Interviews were conducted and tape-recorded with Project Managers, Senior Managers, and Engineering Managers or Technical Leads. The interviews were transcribed and the transcripts were examined as part of our data analysis process.

Stage 2: Change Analysis and Process Evaluation

The collected information described in the Stage 1, were then transferred into spreadsheets. Each change request form was carefully examined. The collected information, such as description of change (requirements change), origin/sources of change, type of requirements (high or low requirements), reason for change, types of change (addition, deletion, and modification), impacted documents, the time when it is raised, and full interval time to process the change, are the main information that were quantitatively analysed. These analyses lead us to better understand the nature of requirements change, its attributes, and its driving factors. This process of analysis required several iterations and the classification of changes were derived inductively.

While analysing the change request forms, we also evaluated the company's process of change management.

Stage 3: Taxonomy Development

Based on the information collected (Stage 2) and our observations, we defined a taxonomy for categorizing requirements change. Our preliminary taxonomy classifies changing requirements based on general types of change, reasons for change, and the change origin. Mapping the changes to the defined taxonomy helps us to determine the causes of requirements volatility. The purpose of this is to improve our knowledge and to better understand the change process and its related activities.

5. Findings

In this section we present our findings in terms of the change process model, the change request arrival rate, the requirements volatility measure, and a taxonomy of requirements change. Finally, we discuss the limitations of the current change management process and causes of requirements volatility.

5.1. Change Process Model

We studied the change management process that was defined by this organization to communicate and manage changes during software product development. This study provided an opportunity to identify problems and to improve the change process.

In this organization, the change management process is driven by change request forms. This represents the locus of information on any change to be made on baseline documents. The change request form is used to request any changes that might impact the project schedule. The change request can be either reports of problems (i.e. bug reports), requests for changes to requirements (addition, deletion, and modification), functionality enhancement requests, or changes to project schedule. This process is the responsibility of a project manager throughout the development life cycle.

We outlined our findings below for the four main phases of the change request process.

Phase 1: Change Request Initialisation

This is the initial phase of the change process where any project engineer or development team members can submit a proposed change and enter the change request (CR) into project database. This phase involves five main activities, which include: identify problem, analyse problem, describe the rationales of the proposed change, perform impact analysis, and fill in change request form.

Phase 2: Change Request Validation and Evaluation

The purpose of this phase is to validate the change request form in terms of the detailed description of the proposed change, its impact on schedule, and the required reviews and approvers to review the change request. The Project Manager is responsible for moderating and managing the change request process. In special circumstances, the project manager solicits and coordinates a discussion with other engineering managers to obtain more information about the impact of the proposed change. When the validation is complete, the project manager circulates the request form to the reviewers and approvers (as stated in the

project configuration management plan) through email. They review the change request in terms of the nature/clarity of proposed change, its impact on project schedule, reasonableness and feasibility of the proposed change.

Phase 3: Change Implementation

This phase starts when the proposed change has been accepted and approved and the change becomes part of the system development. The Project Manager assigns related engineers to implement the change. Communication and coordination among project members is very important because it allows them to trace the change across the impacted products. It is left up to the Project Manager and team members to trace the change.

Phase 4: Change Verification

The objective of this phase is to verify that the change was made correctly. The Verifier (i.e. project manager or quality assurance team) performs verification tasks. If the verification is successful, the initial change request is closed. If it is not successful, the project manager will be notified. In this circumstance, the implemented change needs to be investigated further and change request remains open.

In summary, our analysis identified the following limitations in the change management process: a lack of information about the rationales of the proposed change, the impact analysis of the proposed change has not been performed completely, and the change implementation process is controlled manually.

5.2. Change Request Arrival Rate

This section describes our findings in the change process analysis: the arrival rate of change requests (overall) over time and change requests against requirements throughout the project life cycle.

The arrival rate of overall change requests during development life cycle is presented in Figure 2. There are two legends illustrated in the Figure 2: overall change requests and change requests against requirements. The average rate is relatively low, approximately between two and three change requests submitted per week. In fact, the number of change requests reflects the number of request for requirements change.

Over the course of the project (16 months), a total of 78 change requests were submitted. Most of these requests (86%) were related to product changes, which include changes to requirements specification, functional specification, and software product specification. The rest of the change requests (14%) were related to process/plan changes, which include changes to the project development plan and test plan.

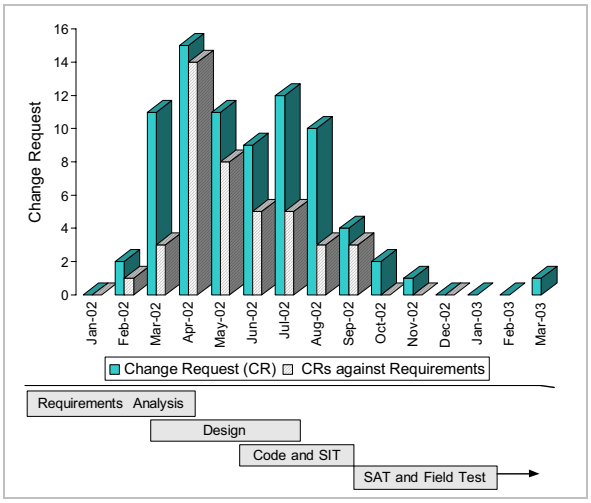


Figure 2. Change requests arrival rate

The rate of change requests increased sharply from March to April 2002 when requirements analysis and documents reviews (i.e. requirements specification, feature proposal, and functional specification) were being completed. During this period, most of the requests resulted in additions and deletions of requirements. This is not surprising, since the developers or engineers received feedback from requirements and functional specification reviews. The arrival of change requests decreased as the project was getting closer to the end of its lifecycle. However the rate of change requests increased again during the end of detailed design review and system integration testing. The majority of change requests during this period related to functional and design specification changes, as the developers gained more knowledge about the product.

5.3. Measuring Requirements Volatility

Since we were only interested in analysing the volatility of requirements, the focus of the analysis was on the change requests that related to changes in requirements and other changes to the software product that affect requirements specification. This section presents our quantitative analysis on the change request data. The objective here is to quantify the extent of requirements volatility throughout system development life cycle.

The measure of requirements volatility is defined as the ratio of the number of requirements change (i.e. addition, deletion, and modification) to the total

number of requirements for a certain period of time (i.e. development phase).

Out of 78 change requests, 42 requests were related to changing requirements. These change requests were carefully examined and evaluated. As a result, we have identified the total number of requirements change throughout the development life cycle and calculated the requirements volatility measure.

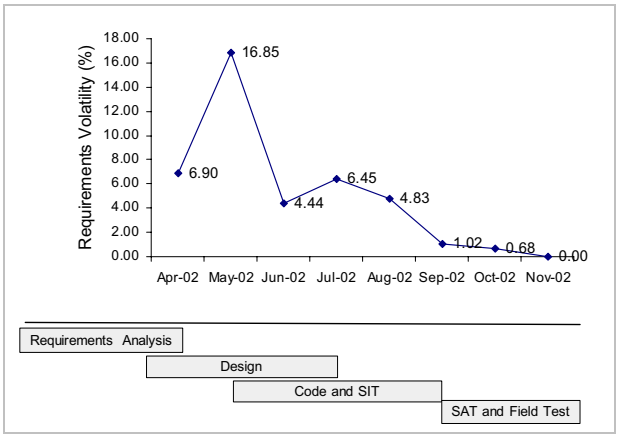


Figure 3. Requirements volatility during development life cycle

The level of requirements volatility at each stage of product development is illustrated in Figure 3. The volatility rate varied across the stages of development and is consistent with the arrival rate of change requests. The overall rate of volatility is 6%, which is considered tolerable. The only high peak (16.85%) was at the end of requirements analysis stage and at the beginning of the design stage. The requirements volatility measure can be viewed as an indicator of how stable requirements are in the system.

Our analysis (as illustrated in Figure 3) indicates that requirements volatility was high at the end of requirements analysis (May 2002). This means that a lot of changes to software requirements occurred in the period when requirements specification reviews were being completed. The volatility decreased sharply in June 2002, however it increased slightly again in July 2002 (at the end of design phase). Then the volatility of requirements decreased steadily at the end of system integration testing and this continued towards the end of the development life cycle.

5.4. Taxonomy of Requirements Change

As part of our analysis, we developed a taxonomy to assist us in understanding the requirements volatility problems. We believe this taxonomy will also allow practitioners or project managers to characterise change requests and improve the change process.

Our taxonomy of requirements change consists of three components: Change Type, Reason, and Origin.

Change Types, is the first component of the taxonomy to classify the change requests in terms of:

- **Requirements Addition**; adding new requirements into the system being developed,
- **Requirements Deletion**; deleting or removing existing requirements from the system,
- **Requirements Modification**; modifying or rewording requirements text.

Reason, this component relates to the categorization of the change in term of the reason or rationales behind the proposed changes. Our classification for the Reason of change is as follows;

- **Defect Fixing**: - changes to correct defects that arise from previous releases
- **Missing requirements**: - requirements were not captured during the initial product definition or discovered after detailed design analysis
- **Functionality Enhancement**: - maintaining or managing functionality for the product releases, e.g. technical upgrade, functionality upgrade, etc.
- **Product Strategy**: - change that related technical engineering and instigated by Marketing group
- **Design Improvement**: - changes that are triggered by improved knowledge of the developers about the product, action items from review documents (i.e. functional specification, design specification)
- **Scope Reduction**: - removing functionality or reducing amount of work due to lack of resources
- **Redundant Functionality**: - unnecessary functionality or functionality that already exists or can be replaced by other existing functions
- **Obsolete Functionality**: - functionality that no longer required for the current release or has no value for the potential users
- **Erroneous Requirements**: - Incorrect or wrong requirements
- **Resolving Conflicts**: changes that triggered by functionality conflicts that exist in the system
- **Clarifying Requirements**: - rewording requirements text for clarification

Origin, is the source of the proposed change, that is, where it originated from. The sources or requirements change could be from: Defect Reports, Engineering's Call, Project Management Consideration, Marketing Group, Developer's Detailed Analysis, Design Review Feedback, Technical Team Discussion, Functional

Specification Review, Feature Proposal Review, and Customer-Support Discussions.

The list of these taxonomy attributes was derived from the change request forms. This taxonomy is our deductive inferences that we used to classify the 42 change requests.

As we mentioned earlier the change request form is a vital element to communicate changes on product deliverables across the project team. Each change request form does not necessarily contain a *single change*, it could contain *multiple changes* that require multiple different actions.

Out of 78 change request forms, we identified 42 change requests that related to changing requirements. We classified these 42 CR according to the three components defined above. We further analysed the data to identify single and multiple change requests. As a result, five (12%) *multiple change requests* and 37 (88%) *single change request*s were identified. It should be noted that the taxonomy was developed based only on single change types and multiple changes described in the next section are not included in the development of the taxonomy.

Change Request with Multiple-changes

Multiple change requests were found to be any of these three combinations: '*addition* and *deletion*, '*addition* and *modification*', or '*deletion* and *modification*' requests (the order is not significant).

Only one of the multiple change requests was of 'addition and *deletion*' combination type. This request was aimed at removing a requirement for a particular operating system that was not supported by third party software in the current release under development. As a consequence of this deletion, a new requirement had to be added to provide an alternative operating system that supported this release. This change request was raised as a result of functional specification review.

A multiple change request of '*addition* and *modification'* combination type was aimed to modify (reword) several requirements due to changes in screen capabilities. As a result of these modifications, new requirements were needed to enable two sub-features exchange the screens definition. This change request was raised as an action from the detailed design reviews.

The last multiple change request we identified is of '*deletion* and *modification'* combination type. The changes involved were as follows:

(1) An obsolete requirement was deleted resulting in modification of several requirements to resolve functionality conflicts. This was raised as a result of technical team discussions

(2) An obsolete requirement was deleted resulting in modification of an existing requirement to address specification changes in data transfer mechanism. This was raised as a result of feature proposal review

(3) A redundant (those that are not necessary or already existed in the previous release), requirement was deleted, resulting in modification of an existing requirement (reword text) for clarity. This multiple change was raised as results of functional specification review

Change Request with Single-change

We classified the 37 (88%) change requests into the three general change types of requirements addition, deletion, and modification. We further classified the data according to the reason category of the changes. Then we linked the changes to their origin or sources.

Mapping the change data to the defined taxonomy attributes enabled us to answer questions of this type: "what are the types of the proposed changes?"; "why is the change needed?"; and "where does the change originate from?". This classification and the relationship of the three components above lead us to better understand the changes and their underlying causes. The following graphs (Figure 4-6) illustrate the relationship of change request attributes resulting from our taxonomy. The numbers on each arrow in these diagrams refer to the number of change requests related to the reason categories or origins.

Figure 4 indicates that the main reasons for adding new requirements were related to improving the design, "functionality enhancement", and "product strategy". The other reasons encountered were: "missing requirements" and "fixing defects" from the previous release. These changes originated mainly from developers/engineers' detailed analysis, feedback from design specification review, marketing group requests, and project management consideration.

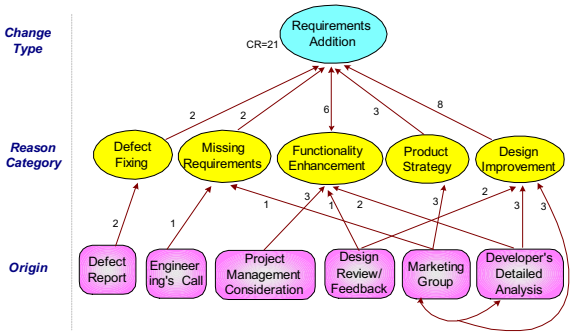


Figure 4. Graphical illustration for requirements addition classification

Adding new requirements in this product release was aligned with the organization's business goals, where functionality enhancement and introducing new functionality are the main concern.

The main reasons for requirements deletion during system development life cycle were to remove 'obsolete functionality" and "requirements redundancy". The other reasons included "erroneous requirements", "scope reduction", and "design improvement". Removing functionality or deleting requirements were originated from marketing group, feedback from design review, and project management consideration. The detailed relationship of these changes is illustrated in Figure 5. In the case of scope reduction, often project management had to consider reducing the amount of work due to lack of resources.

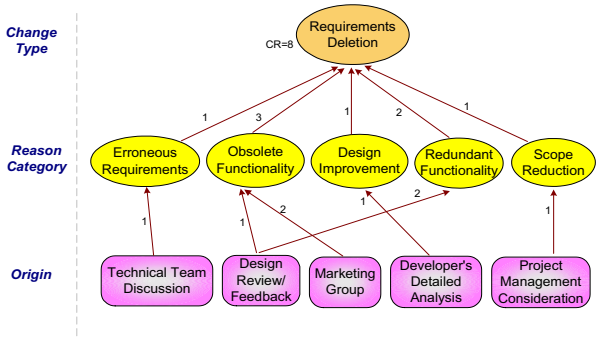


Figure 5. Graphical illustration for requirements deletion classification

The last type of changes is requirements modification, which involves mostly rewording requirements text for clarity and it does not necessarily change the meaning of requirements itself. The main reasons for requirements modifications during system development were to: "clarifying requirements" and "design improvement". The other two reasons for modifications were: "product strategy" and resolving conflicts". The origin of these modifications were mainly from technical team discussions and marketing group requests. The result of our taxonomy on requirements modifications is illustrated in Figure 6.

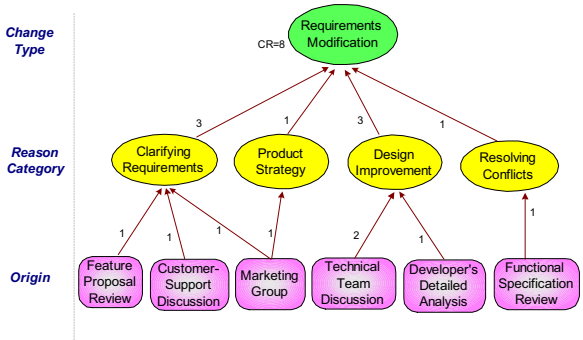


Figure 6. Graphical illustration for requirements modification classification

5.5. Causes of Requirements Volatility

The results of our taxonomy on the change request data provided useful insight to help us draw conclusions about the causes of requirements volatility. It is clearly shown in the three diagrams above that a particular change type has a particular purpose or reason as a result of a specific activity. After carefully examining and analysing the taxonomy of requirements changes in this case study, we identified the root causes of requirements volatility. Three main causes of requirements volatility during system development are:

(1) Changes in market demands, which is a reflection of changes in customer needs

(2) Developers' increased understanding of product domain, which can be explained by most of the requests for design refinements originating from design reviews, technical discussions, and developer detailed analysis, and

(3) Organizational considerations, which is most likely related to the business goals and policy, such as functionality enhancements, product strategy, or scope reduction.

Although our findings regarding the causes of RV is not very surprising and more or less aligns with what is speculated in the literature in various forms, we feel that it increases our understanding of the nature of requirements volatility. Furthermore, this detailed analysis of RV and its causes are very valuable to GDS and other software development organizations that wish to undertake an analysis of their requirements changes.

6. Discussion

The analysis of change request data in GDS has allowed us to develop a comprehensive taxonomy of requirements changes. The main source of our data analysis has been the Change Request forms. The CR form in GDS is currently used primarily as an *operational tool* to allow managers track and communicate changes to software. Our analysis has resulted in an increased understanding of the role that CR forms can and should play in project management. Our study has led us to believe that there are other more important usages of the CR forms that are not currently being considered by most software development organizations such as GDS.

The change request forms, if they contain appropriate information, could be used to contribute to more strategic levels of decision making within the organization. As illustrated in the previous sections, aggregated change request information can be used to assess the nature of requirements volatility.

Furthermore, a taxonomy such as the one developed in this paper could be used as a *strategic tool* to assist project managers in their planning, risk assessment, prediction of effort and cost estimation. For example, if practitioners capture impact analysis data in the change request forms, this information could be used to estimate the effort needed to implement the change more accurately. This is especially effective in developing product line software where the main baseline features remain stable from one release to the other and effort estimates of changes could be carried over from one release to the next with minor modifications.

Although the organization in this case study has implemented change management practices over the last few years, our findings reveal that some activities are still in need of minor improvement.

When we examined the change request forms we discovered that they had little information about the rationale or reasons for the proposed change. The information was inadequate to analyse the importance of the change to be made. We believe this kind information is necessary if we are to analyse problems effectively and understand the proposed changes.

There was no formal impact analysis performed due to inability to predict the potential impact of the change on other related areas. Therefore, it is very difficult for GDS to estimate effort needed to implement the changes at this time. Impact analysis is not a simple task to perform in a large software project. However the benefits of impact analysis are well known in requirements management.

As a result of our case study, some recommendations have been made to GDS management for improving the change request form content as well as the change management process.

7. Conclusions and Future Work

In this paper we have presented the causal analysis of requirements change based on a case study in an industrial setting. The main contributions of this study are twofold. Firstly, a qualitative method for characterizing and evaluating requirements change problems has been developed. This method is described in detail and therefore could potentially be used by other researchers and practitioners in their own environment to identify causes and reasons for requirements changes. Secondly, the analysis of data from change request forms has led us to better understand the nature of requirements volatility during the software development lifecycle and to the development of a comprehensive taxonomy of requirements changes. We have identified the root causes of requirements volatility in a specific project at GDS and been able to offer recommendations on how to improve change management process.

This study is subject to a number of limitations. First, we have conducted a single case study approach. Future work will be undertaken to gather more information about the problem of requirements volatility in other companies. Secondly, our analysis has led us to develop taxonomy of requirements changes in the specific project. This taxonomy needs to be validated. Subsequent stages of this work will investigate the effectiveness of the taxonomy and its benefits in practice from practitioners' perspectives.

This study represents the first phase in a long-term investigation of the phenomenon of requirements volatility and is one of a number of longitudinal investigations currently being undertaken. It helps to set the scene for what is planned to follow. The findings of this case study have provided valuable insight about the dynamic behaviour of software requirements from the beginning of the systems development until the end of the project. The next stage of the research involves developing a model of requirements volatility, its causes and impacts. This model will allow us to identify and develop a set of strategies to manage the impacts of requirements volatility during software development life cycle.

Acknowledgements

We would like to thank all the participants from GDS organisation and the management for their continued support of our research.

References

[1] E. J. Barry, T. Mukhopadhyay, and S. A. Slaughter, "Software Project Duration and Effort: An Empirical Study," *Information Technology and Management*, vol. 3, 1-2, pp. 113-136, 2002.

[2] B. Curtis, H. Krasner, and N. Iscoe, "A Field Study of the Software Design Process for Large Systems," *Comunications of the ACM*, vol. 31, 11, pp. 1268-1287, 1988.

[3] The Standish Group, "CHAOS: A Recipe for Success," 1998.

[4] D. Zowghi, R. Offen, and N. Nurmuliani, "The Impact of Requirements Volatility on Software Development Lifecycle," presented at the International Conference on Software, Theory and Practice (ICS2000), Beijing, China, 2000.

[5] B. W. Boehm and P. N. Papaccio, "Understanding and Controlling Software Cost," *IEEE Transactions on Software Engineering*, vol. 14, 10, pp. 1462-1477, 1998.

[6] M. Christel and K. Kang, "Issues in Requirements Elicitation," Carnegie Mellon University, Pittsburgh TR.CMU/SEI-92-TR-12, September 1992.

[7] G. Stark, A. Skillicorn, and R. Ameele, "An Examination of the Effects of Requirements Changes on Software Maintenance Releases," *Journal of Software Maintenance: Research and Practice*, vol. 11, pp. 293-309, 1999.

[8] Y. Malaiya and J. Denton, "Requirements Volatility and Defect Density," presented at the 10th International Symposium on Software Reliability Engineering, Boca Raton, Florida, 1999.

[9] D. Pfahl and K. Lebsanft, "Using Simulation to Analyse the Impact of Software Requirements Volatility on Project Performance," *Information and Software Technology*, vol. 42, pp. 1001-1008, 2000.

[10 D. Zowghi and N. Nurmuliani, "A Study of the Impact of Requirements Volatility on Software Project Performance," presented at the 9th Asia-Pacific Software Engineering Conference, Gold Coast, Australia, 2002.

[11] S. D. P. Harker and K. D. Eason, "The Change and Evolution of Requirements as a Challenge to the Practice of Software Engineering," presented at Proceeding of IEEE International Symposium on Requirements Engineering, 1993.

[12] W. Lam and V. Shankararaman, "Managing Change in Software Development Using a Process Improvement Approach," presented at IEEE Euromicro Conference, 1998.

[13] D. Rowe, J. Leaney, and D. Lowe, "Defining Systems Evolvability - a Taxonomy of change," presented at International Conference and Workshop on Engineering Computer Based Systems, Jerusalem, Israel, 1998.

[14] J. Chudge and D. Fulton, "Trust and Co-operation in System Development: Applying Responsibility Modelling to the Problem of Changing Requirements," *IEEE, Software Engineering Journal*, vol. 11, 3, pp. 193-204, 1996.

[15] R. K. Yin, *Applications of Case Study Research*, vol. 34: Sage Publications, 1993.

[16] L. C. Briand, V. R. Basili, and Y. Kim, "A Change Analysis Process to Characterise Software Maintenance Projects," presented at International Conference on Software Maintenance, 1994.

PROCEEDINGS

2004 AUSTRALIAN SOFTWARE ENGINEERING CONFERENCE

13-16 APRIL 2004
MELBOURNE, AUSTRALIA

Sponsored by
Australian Computer Society (ACS)
and
Engineers Australia
acting through The Joint Board of the ACS and Engineers Australia

PROCEEDINGS

2004 AUSTRALIAN SOFTWARE ENGINEERING CONFERENCE

ASWEC 2004

13-16 APRIL 2004

MELBOURNE, AUSTRALIA

SPONSORED BY

AUSTRALIAN COMPUTER SOCIETY (ACS)

AND

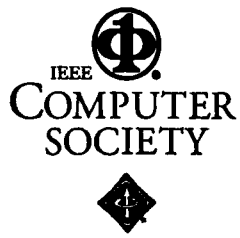
ENGINEERS AUSTRALIA

ACTING THROUGH

THE JOINT BOARD OF THE ACS AND ENGINEERS AUSTRALIA

EDITED BY

PAUL STROOPER



Los Alamitos, California
Washington • Brussels • Tokyo

IEEE Computer Society Order Number P2089
ISBN 0-7695-2089-8
ISSN 1530-0803

Additional copies may be ordered from:

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1 800 272 6657
Fax: + 1 714 821 4641
http://computer.org/cspress
csbooks@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1 732 981 0060
Fax: + 1 732 981 9667
http://shop.ieee.org/store/
customer-service@ieee.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-ku, Tokyo 107-0062
JAPAN
Tel: + 81 3 3408 3118
Fax: + 81 3 3408 3553
tokyo.ofc@computer.org

Individual paper REPRINTS may be ordered at: reprints@computer.org

Editorial production by Frances Titsworth
Cover art production by Joseph Daigle
Printed in the United States of America by The Printing House