Requirements Volatility and Its Impact on Change Effort: Evidence-based Research in Software Development Projects

Return to Published Papers

N Nurmuliani, Didar Zowghi
Faculty of Information Technology
University of Technology, Sydney
{nur,didar}@it.uts.edu.au

Susan P. Williams
School of Business
Faculty of Economics & Business
The University of Sydney
s.williams@econ.usyd.edu.au

Abstract

It is widely acknowledged that requirements volatility (RV) is inevitable and that it causes difficulties during the software development lifecycle. As a major source of risk, the impact of RV is often underestimated; particularly the impact of small changes to requirements in the later stages of the development lifecycle.

This paper presents evidence-based research on RV and its impact on software development effort. The findings are derived from two software project releases within a multi-site software organisation. The paper discusses the usage of a requirements change classification to assist in identifying and assessing the extent of effort required to implement requirements changes. Our research findings reveal that the rate of RV decreases toward the end of the project lifecycle. These empirical findings demonstrate that changing requirements, particularly adding new requirements, at the later phases is considered a high risk because it will cost the organisation in terms of budget overruns or schedule delays.

Keywords: change effort, requirements volatility, requirements change

1. Introduction

It is widely reported that requirements often change during the software/system development process. These changes are caused by several factors, such as errors in original requirements, evolving customer needs, technological changes, and changes in the business environment or organization policy [1-5].

Volatile requirements are regarded as a factor that cause major difficulties during system development for most organizations in the software industry [3]. It is also one of the most important cost drivers [6]. Furthermore, simulation models of software development projects demonstrate that requirements volatility has a significant impact on development effort and project duration [7, 8].

This study is part of our long-term research programme that aims to investigate many facets of requirements volatility [9-13]. In our previous studies we have empirically investigated the occurrence and characteristics of requirements volatility throughout the development lifecycle. In this paper we continue our investigation into the impact of RV on development effort, in terms of total hours to implement requirements changes.

Our previous studies pointed to a need to further explore and examine issues associated with requirements volatility and their costs. Despite the problems of RV being widely acknowledged, its cost in the context of effort has been largely unexplored. Using change request data collected from two software project releases, this paper demonstrates the change analysis approach used to analyse requirements change requests and change effort data.

The following research questions guide the study:

RQ1. What are the types of requirements change that require substantial effort?

RQ2. Which requirements change attributes contribute most to change effort?

This paper is organised as follows. In the next section, we describe the approach used to analyse requirements volatility and its associated cost. Section 3 presents the analysis of the case study findings. Section 4 locates the case study findings in the context of previous related work. Challenges encountered in this study are presented in Section 5. Section 6 provides lessons learned that can be considered important for software researchers and practitioners. Finally, Section 7 provides concluding remarks and directions for future work.

2. Research Approach

This study is part of a longitudinal case study to investigate requirements volatility within software development projects. The purpose of the case study is to empirically analyse requirements change problems in a real software organisation. The study was conducted in a multi-site software development organisation whose development work takes place in Sydney.

2.1. Organisational Context

This case study was carried out at Global Development Systems (GDS)[1] an organisation located in Sydney, Australia. This organisation is an ISO 9001-2000 certified and Capability Maturity Model (CMM) Level 2 software development organisation that belongs to an international multi-site organisation with headquarters and marketing divisions in United State of America (USA). The product strategy is directed from the marketing division, where the development group is located in Australia and New Zealand. Recently, as part of global sourcing, its development group has expanded to India.

The key stakeholder groups or customers of GDS products are scattered across the world. Direct communication between GDS and the customers is minimal. Customer needs are captured by the Business Management (BM) group or Marketing group. These customer needs are then communicated to GDS via Marketing Requirements statements (MR). The communication between GDS and the customer groups is mainly done through BM.

GDS is an engineering laboratory that develops product line software. The product is an enterprise software application generator for mission-critical transaction applications. The software produced is characterised by the delivery of a series of releases. Each release is around 8000KLOC, with development time of between 12-18 months, and involving approximately 130 full time developers.

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

2.2. Data collection

The main source of case study data is derived from change request documents and collected from the organisation's change request database. Other sources of data include informal interviews and discussions with project managers and software developers. This data collection took place throughout the development of software releases (from the end of the requirements analysis phase through to the completion of the system integration testing phase). The detailed data collection and analysis framework for this longitudinal case study has been published in our previous studies [12, 13].

2.3. Data analysis

Change request documents were analysed and relevant change information, such as requirements change attributes and estimated change effort, were extracted.

Content analysis was used to analyse change request forms, by going through each submitted change request form; and identifying substantive statements related to the requirements change attributes, such as the reason for the requirements changes, the types of requirements changes or updates (*addition, deletion, or modification to requirements*) and the sources of requirements change (*change origin*). These attributes are the main components of the requirements change classification developed and reported in our previous work [12, 13]. With respect to the reason for requirements change and change types, an individual estimated effort, in terms of labour hours for each change request, was determined and recorded. At the end of the analysis, the rates of requirements volatility and the estimated change effort are calculated. The correlation between the requirements change attributes and the change effort are also examined.

3. Results and Analysis

The results of this case study are a consolidation of data from two software project releases, *Project Release_A* and *Project Release_B*. Both projects developed similar software applications although they applied different software development methodologies.

The *Waterfall* lifecycle model was used during the development of *Project Release_A*, while an *iterative design and development methodology* was applied for the first time in this organisation on *Project Release_B.*

The findings presented in this paper extend our previous findings [12, 13] in three ways, by enabling us to:

- identify requirements change attributes that potentially contribute to the estimation of the change effort.
- examine the association between requirements volatility and its cost throughout the software development lifecycle
- assess the most expensive requirements change types using the classification we developed

As a result of the analysis of 89 requirements change requests during the development of the two project releases, we identified a total of 266 requirements changes. These requirements are then categorised and assigned using the requirements change classification. Detailed case study findings are described and discussed in the following sections.

3.1. Representing requirements volatility and change effort

Requirements volatility represents the tendency of requirements to change over time. The rate of RV is measured as a ratio of the total number of requirements changes (add, delete, and modify) to the total number of requirements in the system over a period of time. In this study the rate of RV is presented in percentages.

As part of the longitudinal study to improve our understanding of RV issues, this paper focuses attention on the cost of requirements volatility in terms of total effort spent to implement the changes. Mapping the rate of RV and the total change effort over a period of time provides an insight into the most essential information in managing requirements changes. It provides us with an understanding of the magnitude of requirements changes during software development lifecycles.

The result of our change analysis approach on development effort is presented using a requirements change classification scheme. The essential change information we gathered includes: the types of changes (*addition*, *deletion*, or *modification*), the stages in the lifecycle where a change takes place, the reason for change, the complexity of a change (e.g. *number of documents affected*), total number of requirements to be changed, and the source of a change (*change origin*). These attributes are useful for the estimation of the effort to implement the changes. The total effort for each change request was estimated by the originator of the change request and the project managers.

3.1.1. RV and total change effort on *Project Release_A*

Figure 1 illustrates the rate of RV (%) and the estimated effort (hours) during the software development lifecycle for Project Release A. The case study data reveals that the large effort shown in this figure mostly represents the effort needed for additions to the requirements. We found that the later in the development life cycle a new requirement is added, the greater the rework that needs to be done and/or new documents to be produced; thus, more effort is needed. As indicated in Figure 1, the total effort required to implement changes at the later phases of the project's life is higher than in the earlier stages. This finding clearly confirms the conventional view of the cost of requirements changes [14, 15].

Further analysis reveals that most of the effort needed between July 2002 and November 2002 was related to the addition of new requirements. While the rate of RV is high in May 2002, the total development effort in this period is lower than that in July-August 2002. This is a clear indication that development effort is not necessarily proportional to the rate of RV throughout the development lifecycle. The requirements changes occurring later in the lifecycle clearly require more effort due to the proportionally increasing amount of rework.

3.1.2. RV and total change effort on *Project Release_B*

This project experienced volatile requirements towards the end of the lifecycle. Figure 2 shows that the rates of RV are considerably lower during the early phases of the lifecycle (up to September 2003). The changes that took place at this period required less effort to implement them. In contrast, when new requirements are integrated to the project and changes to the existing requirements took place later (October 2003 onward), more effort was needed, particularly at the end of the design phase or during testing. This situation suggests that although RV rates vary and are relatively low toward the end of the project, the cost of implementing these changes remain high. If the changes are unexpected changes and not included in the planned project schedule, this will cause disruption to the project. In the *Project Release_B* case, new requirements were still being introduced during the system integration testing and field testing (January

2004 and March 2004). This clearly can adversely affect the planned schedule.

Overall, Figures 1 and 2, which depict the results of analysing the two variables RV and change effort, reveal that a higher number of requirements changes

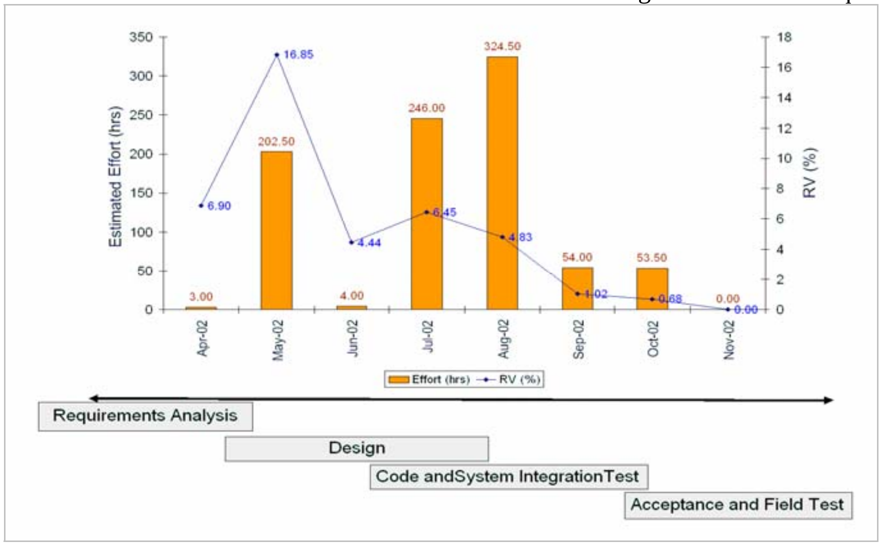

Figure 1. The rate of RV (%) and effort (hrs) throughout the *Project Release_A* lifecycle

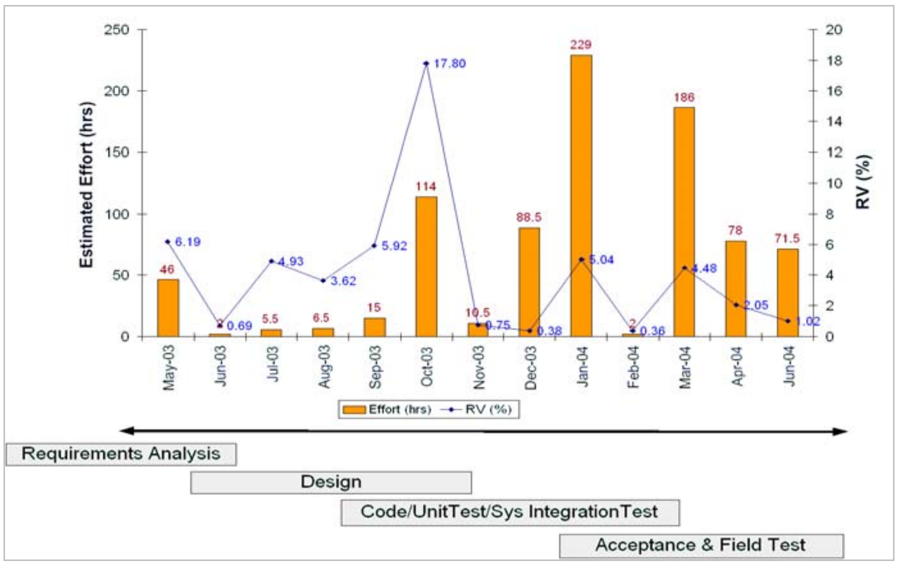

Figure 2. The rate of RV (%) and effort (hrs) throughout the *Project Release_B* lifecycle

does not necessarily imply a higher development effort required to implement the changes. The later in the development lifecycle the changes take place, the more effort will be needed. Although these findings are brought out from only two software project releases, the trend of the cost of requirements volatility confirms the hypotheses resulting from previous studies [14-16].

3.2. Classification and change effort

The following sections present and describe the use of the requirements classification to assess individual change effort and to identify the most expensive types of requirements change. We use two components of the classification: *reason for change* and *change types* to identify the change effort.

3.2.1. Change effort for *Project Release_A*

Figure 3 is a Pareto diagram that illustrates the total estimated effort (hours) attributed to each change categorised by reason and change types. The analysis indicates that the top four changes in the context of high effort (hours) spent were for requirements changes related to '*Functionality enhancement*' (286 hrs), '*Design improvement*' (182 hrs), '*Defect Fix*' (170 hrs) and '*Product Strategy*' (166 hrs). These requirements types, with the exception of '*Defect Fix'*, are the most frequent changes that occurred during the development of this release. A detailed description of the reason for change category is developed and described in our previous studies [12, 13].

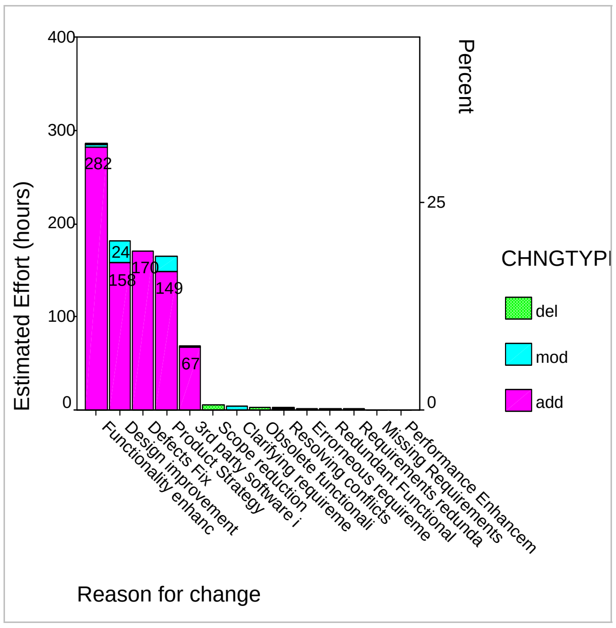

Figure 3. Pareto analysis of total estimated effort - *Project Release_A*

Those changes are also a reflection of the goals[2] of this release. For example, '*Functionality enhancement*' and '*Design improvement*' changes are aimed at maintaining and enhancing existing functionality of previous releases. While GDS has gained a better understanding of its product applications, they also need to improve the current application designs to address new emerging technologies and security related functionality. Similarly, a '*Product Strategy*' related change is aimed to improve the media packaging and licensing of the current release to address the market trend (e.g. Compact Disc and a single license for multiple applications).

Fixing software defects after the product has been released has proven to be expensive. In this case study defects were discovered by the users of the previous release. In order to fix the defects, new requirements were incorporated and substantial effort was required in implementing the change.

Understanding this history of requirements changes and the associated costs is important for future planning, particularly for GDS where software applications are developed in a series of releases.

3.2.2. Change effort for *Project Release_B*

The Pareto diagram (see Figure 4) highlights the top two costs for requirements changes during the development of *Project Release*_B.

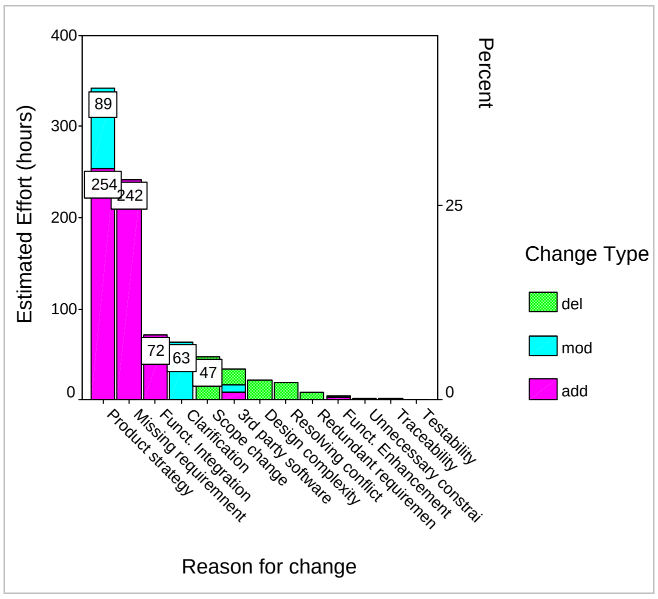

Figure 4. Pareto analysis of total estimated effort - *Project Release_B*

The total highest cost estimated was for requirements change related to '*Product Strategy*' (343 hrs), while '*Missing requirements*' (242 hrs) was the second. Note that '*Product Strategy*' related changes also occurred in this release. This is not surprising since this organisation develops a series of releases in which each release has some similarities with other releases.

'*Missing requirements*' was the second most expensive change type identified during the design phase. Based on the informal interview with GDS's project manager the missed requirements were

[2] The goals were to maintain the application customer base and to enhance its interoperability with new emerging technologies, i.e. .NET and Java-based component environment.

triggered by the inability of the development team to capture the requirements early in the requirements analysis phase. We will revisit this issue in section 6.

The requirements change attributes and change effort data presented in this paper provide valuable information to improve our understanding of the magnitude and significance of each requirements change type. GDS management and software developers are able to use this kind of approach and information to assist them in better handling the consequences of changing requirements during the application development process. In particular they can estimate the effort (hours) required to implement a particular reason-related changes (see Figure 5).

Figure 5 shows the average effort required to implement each change, categorised by the reason for the change. As seen in Figure 5 the average effort required for *Defect* related changes is 85 hours, while the average effort for *Product Strategy* changes is 73 hours. This data provides valuable information for GDS to use in future change effort estimation.

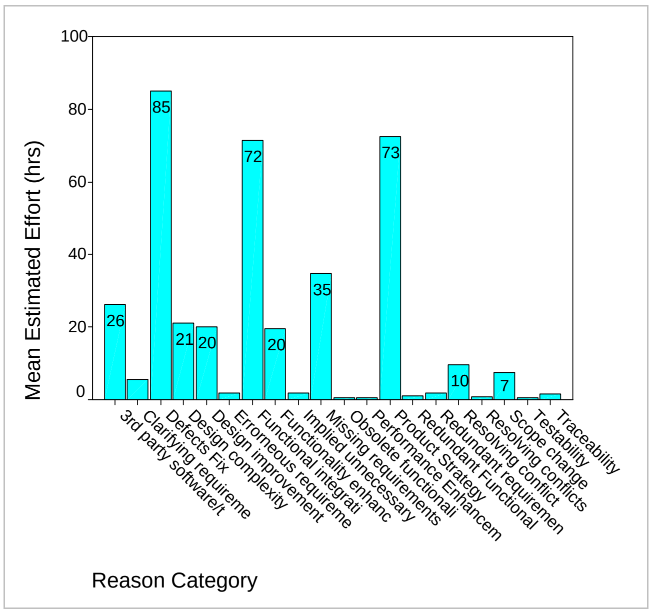

Figure 5. Average requirements change effort by reason category

The findings described here have motivated GDS developers to improve the change effort data collection during change activities by the inclusion of the effort data estimation during the change request initiation. Recommendations have been proposed to GDS regarding an improvement in the change effort estimation model by collecting more data and continuously updating the estimated change effort.

As effort estimation (change effort in particular) was found to be a problem area at GDS, our case study findings serve as a starting point to develop the change effort estimation model for GDS development teams. This paper does not cover the effort estimation model, however it provides directions for future work. Our focus is to identify and assess the potential factors (e.g. requirements change attributes) that may contribute to the estimation of change effort. We investigated the potential association between the attributes and change effort using the *Pearson* correlation coefficient as described in the following section.

3.3. Correlation between requirements change attributes and (estimated) change effort

In this case study, we collected various attributes of requirements changes. These attributes were used to characterise the nature of requirements change. We believe these attributes are useful parameters to estimate change effort during software development projects. The attributes are *total number of requirements changes*, *number of documents affected*, *sources of requirements change (internal and external)*, and *change types (addition, deletion, and, modification)*. These attributes are the most readily available data that we can gather at GDS site.

The statistical test shows that although correlation coefficients between the change effort and each change attribute is less than 0.5 (note: $r > 0.5$ means strongly correlated), all coefficients are statistically significant (see Table 1). This result suggests that those attributes are significant factors to be taken into account for the change effort estimation model.

Table1. Correlation coefficients (r) of change effort and change attributes

Requirements change attributes	**Estimated Effort**
Number of documents affected	**0.384****
Total requirements changes	**0.223***
Sources of change (internal and external)	**0.314****
Change types (deletion, modification, and addition)	**0.404****

**. Correlation is significant at the 0.01 level (2-tailed)
*. Correlation is significant at the 0.05 level (2-tailed)

Undoubtedly the number of documents affected and total requirements changes are significantly correlated to the amount of effort required to implement a change. Interestingly, the sources of change and the change types are also significantly correlated to the change effort.

It is understandable that the sources of the change will determine the number of documents affected and thus determine the effort needed. For example, a requirement change request that originated from the external source (e.g. marketing group/customer) is likely to have impacts on high level requirements, as well as, low level requirements, and other project deliverables. Therefore, the number of documents affected could be high. This potential trend is illustrated in Figure 6. This relationship should be considered when analysing change requests and prioritising their implementation schedule. In light of our findings, the *sources of change* have been incorporated into a new improved change request process at GDS. It is now a mandatory field to be provided in the change request form.

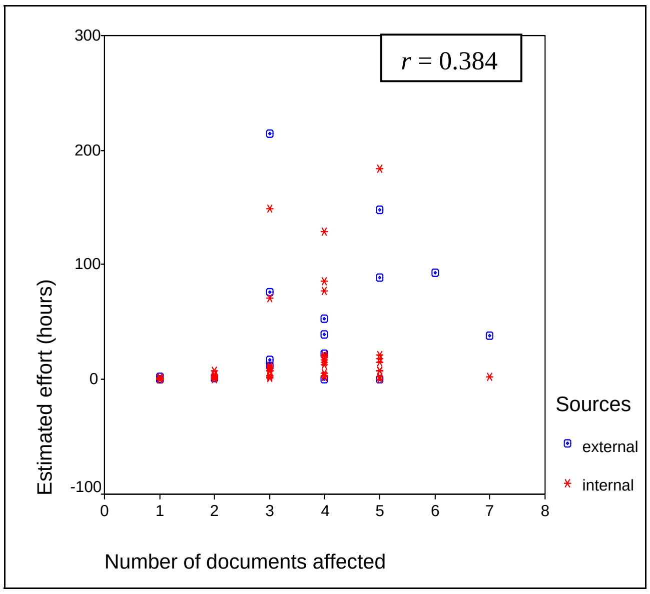

Figure 6. Relationship between change effort and number of documents affected

Additionally, a noteworthy observation from Figure 6 is that the more project documents affected by the change, the more effort is required. Furthermore, the source of the change is a significant factor.

4. Related Work

The work of requirements volatility and its impact on development effort described here closely relates to two areas: classification-related change analysis and requirements change effort.

Classification is one of the most common analysis techniques used to analyse the complex nature of requirements change [17, 18]. In practice, the classification is also used as the basis for software measurement [19]. In this study, the classification is proven to be effective for analysing requirement change requests, particularly in identifying the total cost of requirements changes and the estimated effort for each change category. As an effective technique, requirements change classification can be used to analyse each requirements change request by assessing its sources and its impacts on the software project. As Harker and Eason [20] have suggested, classifying requirements changes is an important factor that needs to be considered in managing requirements change.

It has been reported that increases in RV lead to increases in software project effort. Pfahl and Lebsanft [7] and Ferreira et.al [8] demonstrate in their simulation that RV has a substantial impact on the development effort or overall cost. The results presented here reaffirm the claims of previous researchers. Furthermore we provide evidence of the cost of RV and its magnitude throughout the development lifecycle from our industrial case study data.

Effort estimation is a crucial activity in managing requirements change, particularly in analysing the change impacts [21, 22]. In this paper we demonstrate the factors that are closely associated with the amount of effort to be spent on requirements change implementation. The amount of development effort required to implement the changes will be determined by several factors, such as, total number of documents to be changed, change types (add, del, and mod), the sources of change (external and internal), and total number of requirements changes.

5. Challenges

The industrial case study presented in this paper has provided valuable information and insights that enhance our understanding of requirements volatility within a software development environment. During the case study, however, we encountered several challenges as described in this section. In most cases, to address these challenges the researcher would need to conduct extra data collection activities.

5.1. Change request form was not always complete

Understanding information with regards to a proposed change to requirements is very important. Although GDS has established and implemented change management practices over the last few years, the change request form, which is the sole tool to raise a change to the project deliverables, had little information about the reason for a proposed change.

The information was inadequate for analysing the importance or the reasonableness of the change to be made. This kind information is necessary when we are to analyse problems effectively and understand the proposed changes.

5.2. No formal impact analysis and incomplete change effort estimation

The change information, such as change impacts or change effort estimation, is insufficient in most of the change request forms submitted to the projects. This information is also difficult to consolidate. There was no formal impact analysis performed and the developer (change initiator) was frequently unable to predict potential impacts of a change on some areas. This is possibly due to the complexity of software applications being developed.

Thus, collecting requirements change effort data in this situation is challenging. In order to resolve this problem, an informal interview with the change initiator and project manager was conducted to fill in the gaps in the form.

5.3. Traceability between requirements and other software artefacts (e.g. functional and design specification) was not established

Traceability is an essential technique to support requirements change management, particularly during change impact analysis [21, 22]. The potential impacts of requirements change on some areas can be assessed completely when requirements are traceable to the other software artefacts. In this case study we found that some changes were made to requirements without addressing the potential impacts on other areas. This is because no traceability link was in place to support the impact analysis.

Additionally, the traceability link between requirement change requests and the requirements specification document was not recorded. We found the challenge was significant during the verification of change implementation. Although this organisation used a requirements management tool for recording all requirements, change requests information was rarely captured in the requirements database.

6. Lessons Learned

Despite the challenges encountered, this study provides several lessons that can be considered important for software engineering practitioners and researchers.

6.1. Classification is a strategic approach to understand multiple aspect of requirements volatility

We found that the classification of requirements changes that we developed in this case study and reported elsewhere [12, 13] was useful for the analysis of requirements volatility and its associated costs.

The classification has increased our understanding about the magnitude of individual changes with respect to the change types (add/del/mod) and the reasons for making a change (reason category). In the context of estimated change effort, assigning requirements change requests to the classification helps us greatly in measuring the total cost of requirements changes. In addition, the average of estimated change effort for each change category can be calculated. This lesson extends existing basic principles for managing requirements changes.

6.2. Documenting requirements change information including effort is important

To be able to effectively perform change requests analysis, it is important to record and store requirements change information in the change request database. We have learned that the essential data, such as the purpose of a proposed change, reason for change, change origin, and change types, were required to analyse requirements change both for the purpose of improved understanding and for measuring the rate of change over time.

Estimating effort to implement a change is very important during change request activity because it can assist software developers in analysing, reviewing, and approving a proposed change. In the measurement context, it helps them to calculate the risk of change and the project cost. Furthermore, the historical data of requirements change can be used for estimating individual change effort.

We found that most change requests have little information about change impacts or effort data. A requirement change request should describe its potential impacts on other areas, such as project deliverables or specification documents. Identifying these impacts can lead to the allocation of effort required to implement the change. Following our work several recommendations were made to GDS management with respect to improving the change request form as well as the change management process. These improvements have now been implemented.

6.3. The lower the rate of requirements volatility does not necessarily imply a lower project cost

Our analysis of the data for the two software project releases has helped us to better understand the characteristics of requirements volatility and its costs. The rate of requirements change tends to decrease toward the end of the development lifecycle. However, we found that this does not necessary imply the cost to implement the later change is less, particularly for *Project Release_B* (Figure 2). This is possibly due to the magnitude of individual change that can be explained by the change attributes, such as *change types*, the *reason for change*, or the *sources of change*. From the diagram (Figure 6) and the correlation coefficients (Table 1) we can analyse the relationship between the change effort and the change attributes, and this analysis will lead us to develop a best estimation method for requirements change effort.

6.4. A traceability link between the requirements change request and the requirements document is necessary

Traceability is an important technique during the software development process. Its benefits are widely acknowledged. However, its implementation in practice is rarely followed effectively [23]. We found that the link between a change request identification number and those requirements recorded by a requirements management tool is useful. The link makes it easy for us to track, collect metrics on, and analyse requirements changes. It also helps during the change request verification activity where each approved change request is verified as to whether it has been fully implemented.

6.5. Missing requirements

The comparison of data collected from two releases provides insight into the effectiveness of requirements engineering practices in general, and into requirements elicitation and analysis in particular. It was observed that the development team did not spend as much time and effort in exploring and elaborating requirements earlier in *Project Release_B* as they did in *Project Release_A*. We observed that in *Project Release_B* there were many more missing requirements identified in later stages of its development lifecycle than those in *Project Release_A*.

Furthermore, the software development lifecycle models used were different in the two releases. In *Project Release_A* that used *Waterfall* approach, it seems that RE effort was more focused and performed continuously for the entire release. However, in *Project Release_B*, an *iterative* software development methodology was used where RE effort was somewhat fragmented. This may have contributed to the increase in missing requirements, especially because at GDS, process and tool support for iterative development did not seem to be adequate. It should be noted that the new change control process was not implemented during *Project Release_B* to take advantage of the improvements made to the existing change process and change request form.

7. Conclusions

In this paper we have presented evidence-based research on requirements volatility and its associated costs from two software projects within a multi-site software development organisation.

Analysing the requirements changes has led us to obtain more valuable information in order to better understand the changes and their associated risk. This study has addressed the following research questions:

RQ1. What are the types of requirements change that require substantial effort?

Our analysis suggests that adding new requirements at the later stages of development required substantial effort. Hence, it is the later, very expensive changes that the organisation should pay attention to. In addition to that change types, requirements changes due to *Defect fix*, *Product Strategy*, and *Missing requirements* are considered to be of the expensive change type.

RQ2. Which requirements change attributes contribute most to change effort?

The correlation coefficients indicate that all the change attributes identified in this study, i.e. *total number of requirements change, number of documents affected, sources of requirements change (internal and external)*, and *change types (addition, deletion, and, modification)*, are statistically significant contributors to the change effort.

The major contributions of this paper are twofold: 1) the usage of a requirements change classification to understand the cost of each requirements change, and 2) the relationship between estimated change effort and requirements change attributes.

This study is one of a number of longitudinal investigations currently being undertaken. For the first time, this paper provides empirical evidence about the cost of requirements volatility. The findings have provided valuable insights into the dynamic behaviour of software requirements from the beginning of the systems development until the end of the project. Future work will be undertaken to gather more requirements change data including change effort data (estimated and actual effort) to develop a requirements change effort estimation model.

Acknowledgements

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

References

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

[2] R. Costello and D. Liu, "Metric for Requirements Engineering", *Journal of Systems and Software*, vol. 29, pp. 39-63, 1995.

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

[4] C. Jones, "Strategies for Managing Requirements Creep", in *Computer*, vol. 29, 1996, pp. 92-94.

[5] J. Van Buren and D. Cook, "Experiences in the Adoption of Requirements Engineering Technologies", *CROSSTALK, The Journal of Defence Software Engineering*, pp. 3-10, 1998.

[6] B. W. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[7] D. Pfahl and K. Lebsanft, "Using simulation to analyse the impact of software requirement volatility on project performance", *Information and Software Technology*, vol. 42, pp. 1001, 2000.

[8] S. Ferreira, J. Collofello, D. Shunk, G. Mackulak, and P. Wolfe, "Utilization of Process Modeling and Simulation in Understanding the Effects of Requirements Volatility in Software Development", *in the proceeding of International Workshop on Software Process Simulation and Modeling (ProSim 2003)*, Portland, USA, 2003.

[9] D. Zowghi and N. Nurmuliani, "Investigating Requirements Volatility during Software Development: Research in Progress", *in the proceeding of Third Australian Conference on Requirements Engineering*, Geelong, 1998.

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

[11] D. Zowghi and N. Nurmuliani, "A Study of the Impact of Requirements Volatility on Software Project Performance", *in the proceeding of the 9th Asia-Pacific Software Engineering Conference*, Gold Coast, Australia, 2002.

[12] N. Nurmuliani, D. Zowghi, and S. Fowell, "Analysis of Requirements Volatility during Software Development Lifecycle", *in the proceeding of Australian Software Engineering Conference*, Melbourne, 2004.

[13] N. Nurmuliani, D. Zowghi, and S. Williams, "Characterising Requirements Volatility: An Industrial Case Study", *in the proceeding of International Symposium Empirical Software Engineering*, Noosa - Australia, 2005.

[14] B. W. Boehm and P. N. Papaccio, "Understanding and Controlling Software Cost", *IEEE Transactions on Software Engineering*, vol. 14, pp. 1462-1477, 1988.

[15] Y. Malaiya and J. Denton, "Requirements Volatility and Defect Density", *in the proceeding of the 10th International Symposium on Software Reliability Engineering*, Boca Raton, Florida, 1999.

[16] A. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, A. Ta, and M. Theofanos, "Identifying and Measuring Quality in a Software Requirements Specification", *IEEE- Software Requirements Engineering*, 2nd edition, pp. 141-152, 1993.

[17] I. Sommerville, *Software Engineering*, 7th ed. Harlow, England, New York: Pearson/Addison Wesley, 2004.

[18] G. Kotonya and I. Sommerville, *Requirements Engineering Process & Techniques*: John Wiley & Sons, 2002.

[19] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, 2nd ed: International thomson Computer Press, 1996.

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

[21] R. C. Sugden and M. R. Strens, "Strategies, Tactics and Methods for Handling Change", *in the proceeding of Symposium and Workshop on Engineering of Computer-Based System*, 1996.

[22] K. E. Wiegers, *Software Requirements: practical techniques for gathering and managing requirements throughout the product development lifecycle*, 2nd ed. Washington: Microsoft Press, 2003.

[23] O. Gotel and A. Finkelstein, "An Analysis of the Requirements Traceability Problem", *in the proceeding of 1st International Conference on Requirements Engineering (ICRE'94)*, Colorado Springs, 1994.