A Study of the Impact of Requirements Volatility on Software Project Performance

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

Abstract

Software development is considered to be a dynamic process where demands for changes seem to be inevitable. Modifications to software are prompted by all kinds of changes including changes to the requirements. This type of changes gives rise to an intrinsic volatility, which has several impacts on the software development lifecycle. This paper describes our findings of an extensive survey based empirical study of requirement volatility (RV) and its impact on software project performance. In particular, findings reveal that requirement volatility has a significant impact on schedule overrun and cost overrun in software projects. Our investigation also examined factors that contribute to the extent of requirement volatility and found that variables such as frequent communications between users and developers and usage of a definable methodology in requirements analysis and modeling have impact on the stability of requirements.

Keywords: changing requirements, software project performance, requirements volatility

1. Introduction

Software development is characterized by change. Not only software is considered malleable but also it appears that the deadlines for its delivery to its intended users are often treated as malleable. Recent studies in software development and information systems indicate that large and complex software projects still experience many changes throughout the project life cycle. The evolution of requirements during system development, among other issues, reflects the changing needs of system stakeholders, organization and work environment [1]. Since software development is a dynamic process and characterized by change, software requirements seem to expand while software development is still in progress [2]. As a result, software requirements become increasingly volatile. Software projects often begin with unclearly defined, fuzzy, and incomplete requirements [3][4]. Although an initial set of requirements may be well documented, requirements will change throughout the system development lifecycle. The sources of changes may come from dynamic environments, such as a changing work environment, changes in government regulations, organizational complexity, and conflict among stakeholders in deciding on a core set of requirements [2]. These changes may have significant impact on software project uncertainty, in particular the scope of requirements growth (scope creep).

Although many acknowledge that requirement volatility is one of the risk factors for the success of project, there is no comprehensive empirical evidence that allows us to identify the important factors or diverse issues underlying requirement volatility. Recent empirical studies (e.g. [5] [6] [7] [8] [9]) have investigated the problem of changing requirements and its impact on software development. However, apart from [10] none has specifically focused on factors that contribute to volatility in requirements and on the impact of requirements volatility on software development life cycle.

Previous studies of RV have only focused on the impact of requirements volatility on software productivity [9], software releases [7], or on its impact on isolated phases of development life cycle [6]. Our study focuses on what the previous studies fall short of coverage. That is, investigating factors that may contribute to changing requirements during software development and the impact of requirements volatility on software project performance, in particular software project schedule and project cost.

In this paper we report our findings of a comprehensive survey on requirements volatility and its impact of software project performance. The main objective of this long-term research project is to characterize empirically the detailed impact of requirements volatility, on the diverse characteristics of the software development lifecycle. Evaluating the impacts of requirements volatility on project cost and schedule are among the key issues under investigation. Our aim in the overall study is on investigating the effects of volatility, with the view of constructing a model/theory of requirements volatility and its diverse impacts. This model will become the basis for recommendations on how to improve RE practices in order to effectively manage and control volatility and to assist in reducing effort and schedule variability in practice.

The first research question addressed by this study was: *Is the degree of Requirements Volatility negatively associated with software project schedule and cost performance?* There are three major dimensions that we exploited to investigate and explain requirements volatility. These dimensions are: *potential for change* (changes in business environment), *requirements instability* (the extent of fluctuation in user requirements), and *requirements diversity* (the extent to which stakeholders disagree among themselves deciding on requirements).

The second research question this study addressed is: *What are the requirements engineering practices that contribute to the volatility in software requirements?* This study focused on the issues such as: the usage of definable requirements modeling methods, using requirements management tools, established traceability, performing requirements inspection, the number of user involved in the project, and the frequency of communication between users and development team.

This paper is organized as follows. In the next section we look at the related work on defining RV, and identifying causes and effect of RV. Then the overall research questions and their underlying conceptual model are described. This is followed by the description of data collection and analysis. The next section reports on the details of our findings. The paper ends with discussion and future work.

2. Related Work

2.1. Defining requirements volatility

Although the term 'requirements volatility' is frequently used to express the changing nature of requirements over the system development life cycle, it has not been well-defined in software engineering and information system research literature.

Since software requirements are derived from constantly changing customer's critical business needs, the tasks involved in Requirements Engineering (RE) process are characterized by a high degree of uncertainty. Nidumolu [12] described the *uncertainty* of requirements as the difference in the information needed to identify users needs and the information possessed by the developers. He identified three dimensions of uncertainty: *requirements instability* (which reflects the extent of changes in user requirements in entire project), *requirements diversity* (which reflects the degree of differences among users about requirements), and *requirements analyzability* (the extent to which the process of producing requirements specification can be reduced to objective procedures).

In previous work we have distinguished between two types of requirements volatility: (1) *pre-SRS RV* which refers to the volatility of requirements during the early phases (i.e. elicitation, elaboration, analysis, and negotiations of requirements) of software development and before Software Requirements Specification (SRS) has been completed and signed off, (2) *post-SRS RV* which occurs during the later phases of software development (i.e. design, coding, testing, and deployment) and after the SRS has been completed [10]. We argued that the first type of RV is constructive, while the second type is possibly destructive because it is claimed that it affects the productivity of the software development process, scope creep and the quality of final product.

Recent studies investigated the impact of requirements volatility on software development productivity [9][10][11] and proposed a concept of requirements volatility with respect to the characteristics of software development lifecycle. These studies found that among the proposed dimensions of RV, (such as potential for change in business environment, requirements instability, requirements diversity, and requirements analyzability), only two dimensions were consistent in the analysis, those are requirements instability and requirements diversity.

2.2. Causes of Requirements Volatility

Requirements volatility is a complex phenomenon and no investigation of RV will be complete if the factors that contribute to RV are not considered. There has been a number of factors identified in the literature that cause requirements to change [2][4][7]. These factors include the conciseness of initial requirements definition, the type of system development methodology used, the focus on particular software components, poor communications between users and development team, project size, organizational and environmental factors.

2.3. Impact of requirements volatility

The success of a software project, both functionality and financially, is directly related to the quality of its requirements. Constant changes to requirements during development life cycle significantly contribute to the quality of requirements specification. Further, the unstable requirements have been linked to project risk [5].

Studies of the impact of requirement volatility on software project development lifecycle are rare. Stark et al [7] examined the effects of requirements change specifically on software releases and found that requirements volatility has a major impact on project schedule and cost. Hyatt et al [13] reported requirements volatility must be considered as a part of project risk assessments. Malaiya [6] investigated the relationship between changing requirements and defect density at code phase and found that RV has an impact on defect density.

Lane [9] investigated the impact of RV on effectiveness and efficiency of software development productivity and concluded that there was no direct impact of RV on these two concepts. Lane's finding further suggested that factors such as project size and organization size strongly influence the impact of RV on software development productivity. In our previous works [10][11], we found no strong evidence to suggest that requirements volatility has direct impact on software development productivity (such as code quality, quality of project management and developers capability).

In summary, while recent research has investigated various dimensions of RV in isolation, more evidence needs to be gathered about requirements volatility, its impact and consequences. Given the importance of the phenomenon and the paucity of conclusive empirical research available in this area, it is important to conduct further investigations to better understand the causes and effects of RV, to identify effective processes, techniques and tools to control and manage RV. The results of such studies can be used to develop recommendations to overcome difficulties encountered as a result of volatile requirements. Practitioners need to be aware of such findings in order to cope with requirements change when non-changing requirements are not really an option.

3. Research Methodology

In this section, the conceptual model of the study is presented first. We then describe the survey and data collection methods to test the conceptual model. Finally, we describe the validity of data and the types of data analysis carried out for the model.

3.1. Conceptual Model

The conceptual elements of our research model are presented in a schematic diagram as shown in Figure 1. The model addresses the research questions related to the direct relationship between requirements volatility and software project performance, the influence of project characteristics to this relationship, and the impact of requirements engineering practices to the requirements volatility. The solid arrows indicate direct impact and dotted arrow indicate indirect impact (i.e. control variable)

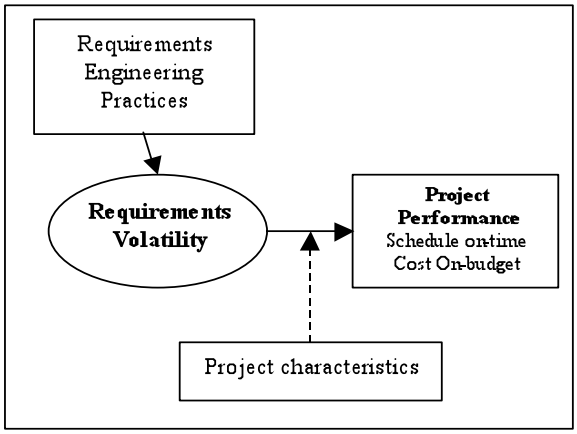

Figure 1 Conceptual Framework

The following variables were used in the analysis:

a) Requirements Volatility. This variable refers to potential for change in business environment, fluctuation in users' requirements (instability), and disagreement among users/stakeholders on requirements (diversity). We developed an 8-item measurement to assess the degree to which volatility in requirements has been established. These items reflect the three dimensions of requirements volatility explained above. We measured this construct using a five-point Likert-type scale that ranged from *strongly disagree* to *strongly agree* for each of the items (Appendix 1).

b) Software Project Performance: Performance of software project is measured by whether the project completed within scheduled and within budget. We specifically asked the respondents the questions as to whether their project completed on-time and/or on-budget.

c) Requirements Engineering Practices: These variables capture several RE practices including: methodology used in modeling and analyzing software requirements, performing requirements inspection, tool used in requirements management, and requirements traceability. These variables together with other variables

(i.e. number of user representatives involved in RE and frequency of communication between users and developers) were examined and analyzed for their contribution to the volatility of requirements.

d) Project Characteristics: These variables cover the specific characteristics of the project under investigation including project size (measured in total development effort, project cost and number of user representatives involved) and organization size (i.e. business turn over).
We investigated the impact of requirements volatility on software project performance, where project performance is defined with respect to the project being completed within schedule and within budget. We also considered that there might be other factors such as project size and organization size that may influence the impact of RV on software project performance. We categorized these factors as project characteristics.

3.2. Data Collection

The research model was tested using a cross-sectional survey of 430 software development companies located across Australia. A survey instrument was designed, pre-tested and sent to the companies' senior IS executives. The target of the survey was recently completed software development projects. The first question we asked the respondents was whether or not they (or their company) are involved in software development. If the answer was no, they did not answer any more questions. If the answer was affirmative, then they were asked to answer the questions with respect to a software project that they have been involved with in the last two years. Our choice of 2-year project duration was based on our experience of a previous study [10]. We recognized that we had to rely on respondent's memory for the answers rather than expecting them to refer to project archives to get the answers to our questions.

A reminder letter was sent 2 weeks after the initial mailing. A total of 92 responses were returned. After careful screening, 40 organizations indicated that they did not develop software in-house, while remaining 52 responses completed the questionnaire. The response rate was calculated by dividing the responses returned by the sample size. The response rate of this survey was 21 percent considered acceptable and is normal for such surveys. As Rogerberg [14] pointed out the typical return rate for a mailed questionnaire is 50% or less, and in recent years it has steadily declined.

Since the response rate is low, it was important to assess the external validity of the study. Non-response bias, if present can be a potential threat to the external validity. To check for non-response bias, we conducted a test on certain characteristic of organization (i.e. firm size). No significant difference was found in firm size between respondents and non-respondents. This suggested that external validity was unlikely to be a problem. In addition, the late respondents were compared against the early respondents. There was no significant difference detected between late and early respondents.

3.3. Data Analysis Technique

Factor analysis followed by a Varimax rotation was used to assess the construct validity of RV item measures. The reliability of scale for each dimension was tested using the Cronbach alpha test. Principal Component Analysis was used to create the key variable, which is requirements volatility. This variable was given by the first principle component of the relevant RV dimensions.

Simple Correlation and Logistic Regression Analysis were used to assess the impact of Requirements Volatility on software project performance. The strength of the relationship between RE Practices and requirements volatility was examined with Multiple Regression Analysis.

4. Findings

In this section we present the main results of our analysis. Starting with the descriptive statistics of the projects across respondents, we summarize the most important variable distributions of the projects. This is followed by detailed results of analysis in relation to the impact of requirements volatility on software project performances. Finally, we discuss the impact of requirements engineering practices on changing requirements in the organizations.

4.1. Descriptive Statistics

A profile of the software projects is summarized in Table 1, while Figure 2 and Figure 3 show the performance of software projects.

Table1. Descriptive Statistics for Project Characteristics

Characteristics	Mean	S.D	Min	Max
1. Project Effort[a]	202	488	1	3,000
2. Project Cost[b]	2,483	7,848	4	45,000
3. Team Size	15	26.64	1	160
4. Users	7	12	1	60
5. Size of software:				
KLOC (n=10)	232	628	3	2,000
FP (n=6)	207	395	1	1,000
Classes (n=8)	123	132	2	400

[a] In person months
[b] In thousand dollars (AUS)
N=52 projects

Figure 2 and Figure 3 illustrate the proportion of projects by cost and schedule performance. These figures summarized the projects that completed or did not complete within their schedule and/or within their budget. It can be seen from both figures that 14% of projects that finished early (i.e. before their schedule) also completed under their budget. On average, most of software projects were completed behind the schedule in range between 25% and 50%, and over-budget between 4% and 25%.

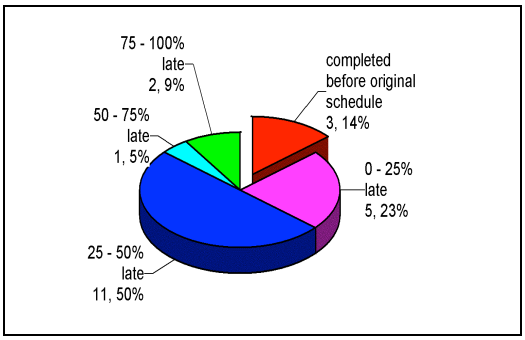

Figure 2 Project Schedule Performance

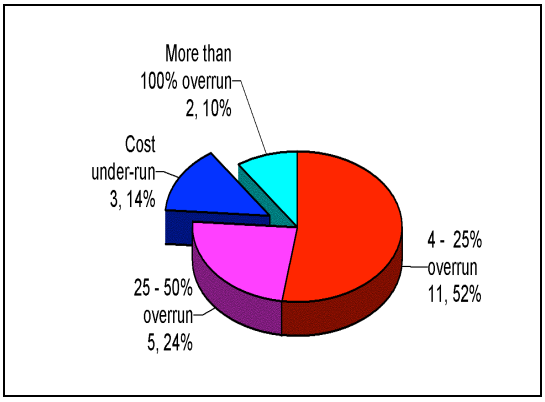

Figure 3 Project Cost Performance

4.2. Construct Validity and Reliability

Construct validity describes the convergent and discriminant validity of the construct's item measures [15] and can be assessed using factor analysis. A factor analysis using varimax rotation resulted in a two-factor solution or dimensions of requirements volatility (see Appendix 2): requirements instability and requirements diversity. An additional dimension (i.e. potential for change) that was proposed initially, did not load significantly in separate construct. Therefore, this dimension was deleted from subsequent analysis. Also, one item from requirements instability dimension was deleted because it loaded less than 0.5 (See Appendix 2 part a).

Reliability analysis provides a measure of the internal consistency of scales. It can be measured by the Cronbach's alpha test, which is treated as correlation coefficient (ranges from 0 to 1), with higher score indicating greater reliability. A value of 0.7 or above is generally recommended [16]. The reliability coefficients of the scales for requirements are given in Appendix1. The test result suggests that the scale reliabilities of the two dimensions are high (greater than 0.7). This means that the scales for these dimensions are reliable.

4.3. Requirements Volatility and its Impact on Software Project Performance

The instability of requirements is characterized by the significant fluctuation of user's requirements in the later stages of development and the differences between requirements that were identified at the beginning of the project and requirements that existed at the end. The latter aspect of requirements volatility is characterized by the differences or disagreement and conflict among users/stakeholders on requirements.

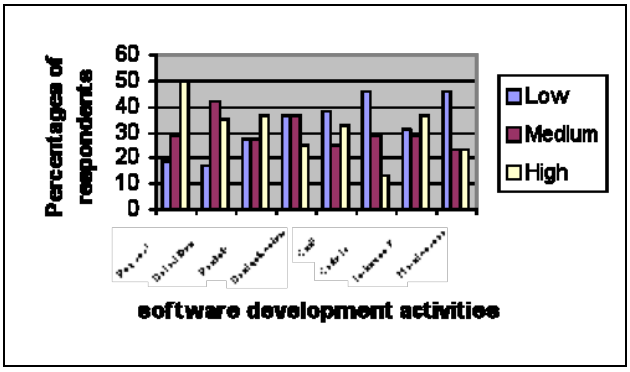

Figure 4 The level of requirements volatility from the respondents' perspective

The respondents were asked to rate the level of requirements change for each stage of software development or during a specific software development activity. The result in Figure 4 shows that most of the respondents indicate the level of changes during requirements analysis is high. This is what we expected because at this stage of software development requirements are being explored, elaborated and fleshed out while new requirements are being discovered as a result of analysis. We have argued elsewhere that this so-called pre-SRS RV is in fact constructive and should be encouraged [10]. At the end of prototyping stage, not surprisingly, the requirements fluctuated as users change their mind after they were shown early prototypes. The

level of volatility seems to be getting lower as the project progresses and near the end of the life cycle, but rose slightly again during code and in-house testing.

The relationship between requirements volatility and software project performances is summarized in Table 2. Software project performance is characterized by whether or not the project completed on time (within original scheduled) and if the final project cost was according to the budgeted amount. The correlation coefficient between requirements volatility and project schedule performance (on time) was negative and significant ($p<0.004$), which gives support to the first research question; i.e. the degree of requirements volatility is negatively associated with project schedule performance. Also, the correlation coefficient between requirements volatility and project cost performance (on budget) was negative and significant ($p<0.05$), which also support the research question: i.e. the degree of requirements volatility is negatively associated with project cost performance.

Table 2. Matrix Correlation

Variables	1	2	3
1. Project on time	1.00		
2. Project on budget	0.30*	1.00	
3. Requirements Volatility	-0.41**	-0.31*	1.00

*Significant at the 0.05 level,
**Significant at the 0.01 level (2-tailed)

Logistic regression analysis was used to test the strength of association between requirements volatility and software project performance measures (i.e. project on time and on budget). In this analysis requirements volatility is treated as independent variable and software project performance (i.e. project on time or on budget) as dependent variable. The reason to use logistics regression analysis was because the response variables were dichotomous (have two values. 1=Yes and 0=No, for the project completed on time or on budget).

The logistic regression model is similar to the linear regression model. The main difference is that the logistic regression model requires the response variable be dichotomous or binary, whereas in linear regression we assume continuous response [17]. Logistic regression is used to predict the binary response variable, using a set of predictors that can be either interval or categorical

We first fitted the logistic model into the data: an independent variable RV and dependent variable project schedule on time. Then we fitted the model of an independent variable RV and dependent variable project cost. The results in Appendix 2 indicate that RV is a significant risk factor that contributes to the performance of software project, the project to meet its schedule and its budget. This is consistent with direct relationships result we found from the correlation coefficients mentioned earlier. These logistics regression estimates indicated that an increase in requirements volatility resulted in decreasing the probability of project to complete on time as well as on budget.

4.4. Requirements Engineering Practices and its Relationship to Requirements Volatility

In this study we were interested to investigate which of the current practices in requirements engineering (RE) may contribute to volatility in requirements. We asked the respondents specific questions regarding various aspects of the RE practice. These include the use of a specific methodology in analyzing and modeling requirements, performing requirements inspection, use of requirements management tool, and extent of requirements traceability. These variables together with other variables (i.e. number of user representatives in the project and frequency of communication between users and developers), and their contribution to requirements volatility were analyzed. Multiple regression analysis was used to test the relationship between dependent variable requirements volatility and multiple independent variables (i.e. the variables related to requirements engineering practices mentioned above). We examined linearity, homoscedasticity, independence of the residuals and normality of the data, and we found the data did not violate the assumptions underlying multiple regression analysis. The regression estimates are summarized and shown in Table 3.

Table 3. The impact of RE Practices on Requirements Volatility (OLS Regression Estimates)

Independent Variables	Require-ments Volatility
Usage of RE methods(1=yes, 0=No)	**-0.827****
Perform Reqs. Inspection (1=yes, 0=no)	**-0.770***
Traceability in place(1=yes, 0=no)	0.438
Usage of RM tool (1=yes, 0=no)	0.823
User representatives (Logarithm)	**0.512****
Frequency communication (Logarithm)	**-2.096*****
R^2	0.556
Adjusted R^2	0.444
F	**5.000*****

* Significant at the 0.1 level,
** Significant at the 0.05 level,
***Significant at the 0.01 level (1-tailed)

Table 3 shows the results of multiple regression analysis for the effect of RE practices on requirements

volatility. The practices such as a definable methodology used in modeling and analyzing requirements, frequency of communication between users and developers, and performing requirements inspection had significant negative impact on requirements volatility. Number of user representatives had a significant positive effect, while established requirements traceability and the usage of requirements management tool also had positive coefficients, though they were not statistically significant. We found only 44.4% (Adjusted R^2) of the variation in requirements volatility (increases or decreases) can be explained with those significant RE practices. Although requirements traceability and the use of requirements management (RM) tools can be removed from the model due to statistical insignificance, we concede that they are both important issues in requirements management. It is commonly believed that having effective traceability in place and using a RM tool enhances requirements management practices significantly [19].

5. Discussion

We now revisit the research questions posed in the introduction and discuss the findings.

The findings indicate that there is a negative relationship between requirements volatility and software project performance, measured by project completing on time and on budget. There is a clear indication that the more unstable requirements become the more likely it is that the project will be completed behind schedule and over budget. We contend that requirements volatility is not the only factor that would affect the project delays or project cost overruns. We felt that additional investigation was required to examine the influence of other factors. Hence, we tested the impact of factors (as control variables) such as organization size (i.e. business turn-over) and project size (i.e. project cost, total effort, and number of user). We added these control variables into the logistic model. However, the regression analysis suggested none of the control variables were significant.

For the second research question under investigation in this study, we examined the impact of several RE activities and related issues of requirements practice on RV. The findings indicate that using a definable methodology for requirements analysis and modeling has negative impact on RV. This is exactly what we expected since the stability of requirements bound to be affected by rigorous procedures imposed on analyzing requirements. The specific question we asked here was:

> Which of the following approaches did you use in analyzing and modeling software requirements:
>
> a) structured analysis and design techniques,
> b) b) object oriented analysis,
> c) c) no methodology".

Interestingly, 38% of respondents indicated they do not use a methodology. We wondered why that is the case but this is a question that could not be answered from this survey and we leave it for future study.

Another activity of RE process that we investigated with respect to the second research question was the impact of performing requirements inspection on RV. The findings indicate that performing requirements inspection reduces the extent of RV. Formal requirements inspection are claimed to be a cost-effective technique to discover requirements defects [17]. However, because inspection needs people from different skills and organizations to get together, it is reported that most project teams do not carry our formal inspection of requirements before design begins. We believe that this is an important finding for motivating practitioners to spend adequate time for validating requirements.

Frequency of communication between developer and customers was yet another issue that we investigated in relation to its impact on RV. The findings suggest that the more frequent developers and customers communicate with each other during RE, the less volatile their requirements will be. A related issue under investigation was to do with the number of user representatives involved in the development team. The findings suggest that the more user representatives involved in the development team the more volatile the requirements were. These two contributing factors seem to be competing and we feel that it requires further investigation. The choice of whether to have more meetings with the stakeholders versus involving a number of users in the development team is not an easy one to make and we leave it for future work to investigate this issue possibly with case studies.

Finally the other two related factors we examined were the impact of using requirements management tools and established traceability. The findings did not suggest any significant impact on requirements volatility.

6. Conclusion

In this paper, we have reported the results of a survey-based study of the impact of requirements volatility on software project performance as well as the impact of various RE practices on requirements volatility.

Studying requirements volatility provides an important insight for software developers, so that they can pay due attention to RV and its associated risks. This increased understanding will then allow them to control and manage RV as well as reducing its potentially damaging impact on software development process and product. This will further facilitate a more effective handling and management of the impact of RV thus resulting in better estimates, better risk management and

better quality software while reducing schedule overruns and cost overruns.

In that this study is one of a number of longitudinal investigations of requirements volatility, as part of a long-term investigation of this phenomenon, it helps to set the scene for what is planned to follow. The next stages of the research involve *additional analysis* of the survey questionnaire data and specifically targeted *industry interviews and field studies*, both with the aim of refining our understanding of impacts of requirements volatility, especially from the point of view of establishing verifiable causal relationships.

Appendix 1. Measurement Details for Requirements Volatility Constructs

Response Scale:

"*Please indicate what degree you feel the following statements describe the volatility of software requirements"*

Five-point scale: 1=strongly disagree, 3=neutral, and 5=strongly agree.

Potential for Change (Cronbach's Alpha = 0.28)

1. The changes in business environment for this project were high.
2. The analyst's knowledge of the business environment was excellent.

Requirements Instability (Cronbach's Alpha = 0.74)

1. Requirements fluctuated in the earlier stages
2. Requirements fluctuated in the later stages
3. Difference in requirements identified at the start of the project from the final requirements

Requirements Diversity (Cronbach's Alpha = 0.77)

1. It was difficult for stakeholders to reach agreement among themselves on requirements.
2. A lot of effort had to be spent in incorporating the requirements of various users
3. It was difficult to customize software to one set of users without reducing support for other users

Appendix 2. Statistical Analysis

a. Factor Analysis output

Measurement Items	Instability	Diversity
1. Business-change1*	.867	-.090
2. Business-change2*	.315	-.507
3. Instability1*	.173	.216
4. Instability2	**.744**	.096
5. Instability3	**.742**	.447
6. Diversity1	.390	**.729**
7. Diveristy2	.001	**.882**
8. Diversity3	.497	**.600**
Eigenvalue	2.384	2.178
% of total variance	52.3%	48.0%

* items deleted for subsequent analysis

b. Logistics Regression Estimates

Independent variable	Response variables	
	Project on time	Project on budget
Requirements Volatility		
β	-1.001	-.682
Exp(β)	0.368	0.043
S.E	0.367	0.337
p-value	*0.006*	*0.043*
Constant		
α	0.394	0.192
p-value	0.209	0.538
Goodness of fit test (Chi-square)	5.786 *(0.67)*	2.885 *(0.89)*
N	52	46

References

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

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

[3] H. Krasner, "Requirements Dynamics in Large Software Projects," proceedings of *the 11th World Computer Congress (IFIP89*), Amsterdam, The Netherlands, 1989.

[4] M. Christel and K. Kang, *Issues in Requirements Elicitation*, Carnegie Mellon University, Pittsburgh September 1992.

[5] T. Hammer, L. Huffman, and L. Rosenberg, "Doing Requirements Right the First Time," *CROSSTALK, The Journal of Defence Software Engineering*, December 1998, pp. 20-25

[6] Y. Malaiya and J. Denton, "Requirements Volatility and Defect Density," proceedings of *the 10th International Symposium on Software Reliability Engineerin*g, Fort Collins, 1998.

[7] G. Stark, A. Skillicorn, and R. Ameele, "An Examination of the Effects of Requirements Changes on Software Releases," *CROSSTALK, The Journal of Defence Software Engineering*, December 1998.

[8] 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.

[9] M. Lane and A. L. M. Cavaye, "Management of Requirements Volatility Enhances Software Development Productivity," proceedings of *the 3rd Australian Conference on Requirements Engineering (ACRE 98*), Geelong, Australia, 1998.

[10] D. Zowghi, and Nurmuliani, "Investigating Requirements Volatility During Software Development: Research in Progress", *Proceedings of the 3rd Australian Conference on Requirements Engineering (ACRE98)*, Geelong, Australia, 1998

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

[12] S. Nidumolu, "Standardization, Requirements Uncertainty and Software Project Performance," *Information & Management*, vol. 31, pp. 135-150, 1996

[13] L. Hyatt and L. Rosenberg, "Software Metric for Risk Assessment," proceedings of *the 26th Safety and Rescue Symposium, Risk Management and Assessment Session*, Beijing, China, 1996.

[14] S. G. Rogelberg and A. Luong, *Nonresponse to mailed surveys: A review and guide*, *Current Direction in Psychological Science*, vol. 7, pp. 60-65, 1998.

[15] K. A. Bollen, *, Structural Equations with Latent Variables*, Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, Inc. NY, 1989

[16] J. C. Nunnally, *Psychometric Theory*, McGraw-Hill, New York, 1978

[17] D. McNeil,, *Epidemiological Research Methods*, Wiley, 1996

[18] I. Sommerville, P. Sawyer, *Requirements Engineering, A good practice guide*, Wiley, 1997

[19] K. E. Wiegers, "Software requirements", Microsoft press 1999.