A Correlational Study on Four Measures of Requirements Volatility

Annabella Loconsole
Department of Computing Science,
Umeå University, Sweden,
bella@cs.umu.se

Abstract

Requirements volatility is an important risk factor for software projects. Software measures can help in quantifying and predicting this risk. In this paper, we present the results of a correlational study with the goal of predicting requirements volatility for a medium size software project. Based on the data collected from two industrial software projects for four measures of size of requirements (number of actors, use cases, words, and lines), we have evaluated prediction models for requirements volatility. These models can help project managers to estimate the volatility of requirements and minimize the risks caused by volatile requirements, like schedule and cost overruns. In cross systems validation our best model showed a mean magnitude of relative error (MMRE) of 0.25, which can be considered reliable. In an earlier study, we showed that decisions solely based on developers perception of requirements volatility are, instead unreliable. Predictions models, like the ones presented here, can therefore help taking more reliable decisions.

Keywords: Requirement, Prediction Model, Empirical Validation, Correlational Study

1. INTRODUCTION

High requirements volatility can cause cost and schedule overruns, making the goals of the project hard to achieve. Studies show that requirements volatility has a high impact on project performance [16, 19]. Since we cannot expect requirements to be stable, even when requirements engineering tasks are performed well, we should at least carefully monitor them throughout the software life cycle. Monitoring requirements volatility usually involves measuring trends or percentages of changes to requirements, and quantifying and predicting changes to requirements. By anticipating a certain level of volatility, project managers can take appropriate actions in order to decrease project risks. For instance, they can assign extra resources to critical requirements, postponing delivery dates for the part of the software system including critical requirements and re-estimate the overall cost.

In two industrial empirical studies [12, 13], the authors investigated measures of volatility for a medium size software project. The data analysis showed a high correlation between each of the size measures and the total number of changes. This suggests that the measures of size of requirements documents are good indicators of the number of changes for (use case-based) requirements documents. Furthermore, the data analysis did not show any significant correlations between any of our four volatility measures and the rating of volatility by the experts.

Based on those results, we have performed a correlational study with the goal of empirically validating four measures of size as predictors of requirements volatility. Here we describe the results of that study, further details can be found in [12]. We built four prediction models using data collected for a medium-size software project developed at BAE Systems Hägglunds AB, Sweden. We then evaluated the accuracy of one model by applying it on a set of data collected for a second project at the same company. The results show that the size measure "number of lines" (of a requirements document) is a good predictor of volatility. Other size measures (number of actors, number of use cases, number of words) were not found to be significant predictors.

TABLE 1: Key data of the two projects analysed

	Project A	Project B
Number of developers	10 (incl. 2 proj. managers)	15
Project duration	30 months	48 months
Use case documents (UCD)	14 files	22 files
Average UCD size in kB (Median)	60	76.5
Total number of req. documents	16	26
Vision/overview documents	1	2
Supplementary req. documents	1	2
Total number of use cases	39	22

The present work is unique regarding two aspects. First, we aim to predict requirements volatility, while volatility is usually chosen as an independent variable, i.e., as predictor of other software or project attributes, like for example in [1, 10, 19]. Second, we are concerned with volatility of smaller units of requirements, instead of treating volatility as a property of the whole set of requirements of a project. This gives project managers a more fine-grained tool for requirements management.

The remaining part of the paper proceeds as follows: section 2 describes the context of the study, goals, hypotheses, the measures used in the study. The construction and validation of the prediction models is described in section 3. Finally, discussions and conclusions are presented in section 4.

2. DESCRIPTION OF THE PRESENT CORRELATIONAL STUDY

The goal of the present study is to analyse the ability of four specific measures to predict the volatility of requirements, using two data sets from two different projects (see table 1). The hypothesis of the study is the following: the size measures NACTOR, NUC, NWORD, and NLINE are good predictors of requirements volatility. Our hypothesis is built on the idea that larger requirements are affected by changes more than smaller ones, because they contain more information. Although quite intuitive, this relationship has not been scientifically proven yet.

Prediction models were constructed applying linear regression analysis using the data set from project A. The best model was then validated using the dataset from project B. We choose linear regression because it is suggested to predict interval and ratio scale dependent variables [5], which is our case. Data collection was semi-automatic, carried out by the authors by studying the documentation of the projects A and B. From the first available revisions of requirements documents, all files were analysed following the rules described in [12].

2.1. Context of the study

We analysed and collected data from the use case-based requirements specifications¹ of two different software projects performed at BAE Systems Hägglunds AB, Sweden (see table 1). The company produces automotive systems with embedded software and is ISO9001 and ISO14001 certified. At the time of the analysis the software systems had been in operation for approximately 24 months. The Rational Unified Process (RUP) was used in both projects. The goal for project A was to develop external diagnostics software for personal computers. While Project B developed an information and control system for the vehicles constructed by the company. A brief description of the projects analysed is shown in table 1. Further details about the projects can be found in [12].

2.2. Independent variables

The entities analysed in the two projects were requirements documents. Intuitively, the larger the document the more changes there are. Therefore, we believe that the size of requirements is an important factor affecting volatility. The size measures “number of actors interacting with the use cases described in the file” (NACTOR), “number of lines per file” (NLINE), “number of words per file” (NWORD), and “number of use cases per file” (NUC), are the independent variables chosen for this study. These measures were calculated by the authors using a computerized tool. Although the measures NLINE and

¹We are aware of the fact that some researchers do not consider use cases to be requirements [18, 20].

TABLE 2: Univariate regression analysis for data set A ($\alpha = 0.05$)

Model	Measure	Coeff.	Std. Err.	P-value	R ²
1	NACTOR	0.3258	0.1241	0.022	36.5
2	NUC	0.08278	0.05566	0.163	15.6
3	NLINE	0.007133	0.002504	0.015	40.3
4	NWORD	0.0005351	0.0003324	0.133	17.8

NWORD are very similar, we collected data for both measures to compensate for possible differences caused by formatting and style.

Another possible choice for the independent variable could have been use case points (UCPs) [2, 18], “number of dependencies between use cases” or “number of steps in scenarios”. However, these measures are highly dependent on the use case format used in a particular project. If use cases are described in plain text only (the most basic format), this information might not be available. Therefore, we discarded such measures for the sake of generality.

2.3. Dependent variables

To investigate the relationship between requirements size measures and requirements volatility one needs a suitable and practical measure of volatility as the dependent variable of our study. Theoretical definitions of requirements volatility are presented in [3, 15, 17], while operational definitions can be found in [3, 13, 15, 17, 19]. Baumert and McWhinney suggest measuring source and state of change [3], while Nurmuliani [15] take into consideration the source of change in their theoretical definition of volatility. All other definitions are purely quantitative, i.e., they focus on the amount of changes (additions, deletions, and modifications) to requirements and do not consider the cause of change and the semantics of a change, i.e., in what way a particular change impacts development. Likewise, we define requirements volatility as the amount of changes to a requirements document over time and measure it as the sum of the change densities of a requirements document.

$$Volatility = \sum_{i=1}^{NREVISION} \frac{NCHANGE_i}{NWORD_i}.$$

Our operational definition of requirements volatility is a function of number of changes (NCHANGE), time measured in number of revisions (NREVISION), and size of the requirements document measured in NWORD. NCHANGE is a count of changed words, therefore, NWORD was chosen to calculate the change density (having the same unit of measurement). NREVISION is a count of the revisions for a file, where a revision is a version of a file with a unique identifier. Our dependent variable has been determined by counting the changes from one version of a document to the next by means of a tool.

There is an important difference between our operational definition of volatility and the ones described in the literature. While common definitions consider volatility as a property of all requirements of a project, we look at volatility document by document. This more fine-grained view of volatility makes it possible to distinguish units of requirements that are particularly volatile from those that are more stable.

3. CONSTRUCTION AND VALIDATION OF PREDICTION MODELS

In our previous studies, we found a strong correlation between all four size measures and total number of changes. Based on those results, we have analysed the ability of our measures to predict the volatility of requirements. The data analysis was obtained by following the procedure suggested by [5, 6]. In [12] we described the data sets A and B, the principal component analysis, univariate regression analysis, multivariate regression analysis, and sanity tests on the regression models. We applied linear regression to each of the four measures (also called ordinary least squares regression), which is most suitable to predict a dependent variable at the interval or ratio scale [5]. Two of the four measures had a statistically significant positive relationship with the dependent variable Volatility (see table 2). After performing the sanity checks, only model 3 (NLINE) was considered for the model validation.

TABLE 3: Model Validation

	Model	Measure	MMRE	MdMRE	Pred(0.25)	Pred(0.4)
Evaluation of the goodness of fit (data set A)	3	NLINE	0.14	0.1255	0.93	1
Prediction model validation (data set B)	3	NLINE	0.25	0.21	0.68	0.86

3.1. Evaluating goodness of fit

In this section, we evaluate the goodness of fit of model 3. The measures we used to evaluate the accuracy of the prediction models are the mean magnitude of relative error (MMRE), the median magnitude of relative error (MdMRE), and the threshold measure $\text{pred}(n)$. Our model 3 has $\text{pred}(0.25) = 0.93$, which is quite reliable according to [8]. Since there are no other prediction models of volatility, we cannot compare these results with prediction models from exactly the same application domain. However, considering other areas, like object-oriented design or software maintenance, the goodness of fit of our model is better than those presented by Genero et al. [9] (whose best model has a $\text{MMRE}=0.24$) or by MacDonell [14] (whose best model has a $\text{MMRE}=0.21$). Similarly to our case, these models were evaluated on the same data set they were constructed. We would therefore expect to get high accuracy. To evaluate model reliability in a more realistic way, we need to apply it to data from other projects, which is done in the next section.

3.2. Prediction model validation

Our situation is ideal for evaluating prediction models, since separate data sets are available that have been derived from different projects, but within similar environments. The prediction model is built from one data set and then used to make predictions for another project. Model 3 showed good reliability when applied to data set B (see table 3). $\text{Pred}(0.25)$ is almost in the range considered as reliable by De Lucia [8], who admit that $\text{pred}(0.25) > 0.7$ is difficult to achieve. The values of MMRE and of MdMRE are also in the range of the recommended values for reliable models. Our model performs better than other cross validated models, like for example COCOMO [4] ($\text{MMRE} = 0.6$ and $\text{pred}(0.25) = 0.27$) or Jorgenssen's best model ($\text{MMRE} = 1.0$ and $\text{pred}(0.25) = 0.26$) [11].

4. DISCUSSIONS AND CONCLUSIONS

In this paper, we have described the results of a correlational study on requirements volatility performed at BAE Systems Hägglunds AB, Sweden. We collected and analysed data of two historical projects in the company for the measures NUC, NACTOR, NLINE, NWORD, NREVISION, and NCHANGE. Applying univariate and multivariate regression analysis, we built prediction models using data collected on a medium size software project. Only the model having NLINE as covariate was found significant. We then evaluated the model accuracy by applying it on a set of data collected on a second, slightly larger, project developed at the same company. The model showed good performance when applied on data set B, the values of MMRE obtained are in the range of the recommended values for reliable models. Our prediction model receives as input the number of lines of a file describing software requirements. The model produces a number as output: the sum of the change densities in time for this requirements document. By regularly comparing the number obtained (the predicted volatility) to the current volatility (as suggested by [7]), project managers can identify critical requirements and allocate resources for analysing reasons for their volatility. In this way they can minimise the risks of schedule and cost overruns.

In the present work, we deal with two factors of volatility: the number of changes to requirements and time. Volatility is a quite complex concept depending on many more factors than size. For a deeper analysis of volatility it is suggested to investigate qualitative aspects such as why the changes occur, how critical the changes are, the type and phase where the changes occur. Regularly investigating many qualitative aspects on many requirements documents is expensive and subjective and therefore not feasible. However, studying the impact of a change might help to "classify" changes and to identify the most critical changes. In a qualitative analysis, we would care only about important changes since all others will not affect the project much. With our prediction model we can quantify the "instability" of requirements in order to identify the critical ones. When the critical requirements are identified, we can perform a deeper analysis of the changes in order to figure out the problems with the requirement.

Although our approach currently uses use case-based requirements, the defined measures are quite general and can therefore be used for all use case documents written in textual form. Furthermore, the measure NLINE can be applied to any kind of requirement written in textual form. Our approach of using counts is simple, effective, easily interpreted, and completely automated.

We believe it is possible to develop prediction systems based on simple measures such as ours. The reason for our confidence is that the correlational study presented here is based on two different industrial projects. Although the exact nature of the prediction systems will vary from company to company, the underlying principle is the same. That is, developers can collect data for simple measures derived from requirements documents, and build effective prediction systems using techniques like linear regression analysis. Models must however be calibrated to suit different environments [14].

REFERENCES

- [1] AMBRIOLA, V., AND GERVASI, V. Process metrics for requirements analysis. In *Proc. of the 7th European Workshop on Software Process Tech.* (Feb. 2000), pp. 90–95.
- [2] ANDA, B. Comparing effort estimates based on use case points with expert estimates. In *Empirical Assessment in Software Engineering (EASE)* (Keele, UK, 2002).
- [3] BAUMERT, J., AND MCWHINNEY, M. *Software Measures and the Capability Maturity Model*. Technical Report CMU/SEI-92-TR-025, Pittsburgh, July 1992.
- [4] BOEHM, B. *Software engineering economics*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.
- [5] BRIAND, L., AND WÜST, J. *Advances in computers, chapter: Empirical studies of quality models in object-oriented systems*. Academic Press, INC, USA, 2002.
- [6] BRIAND, L., WÜST, J., DALY, J., AND PORTER, D. Exploring the relationships between design measures and software quality in object-oriented systems. *J. of Sys and Soft.* 51, 3 (2000), 245–273.
- [7] COSTELLO, R., AND LIU, D. Metrics for requirements engineering. *J. of Sys and Soft.* 29, 1 (Apr. 1995), 39–63.
- [8] DELUCIA, A., POMPELLA, E., AND STEFANUCCI, S. Assessing effort estimation models for corrective maintenance through empirical studies. *J. of Inf. and Soft. Tech.* 47 (2005), 3–15.
- [9] GENERO, M., PIATTINI, M., MANSO, M., AND CANTONE, G. Building UML class diagram maintainability prediction models based on early metrics. In *Proc. of the 9th Int. Metrics Symp* (2003), pp. 263–275.
- [10] JAVED, T., MAQSOOD, M., AND DURRANI, Q. S. A study to investigate the impact of requirements instability on software defects. *SIGSOFT Softw. Eng. Notes* 29, 3 (2004), 1–7.
- [11] JØRGENSEN, M. Experience with the accuracy of software maintenance task effort prediction models. *IEEE TSE* 21, 8 (Aug. 1995), 674–681.
- [12] LOCONSOLE, A. *Definition and Validation of Requirements Management Measures*. PhD Thesis, Technical Report UMINF 07.22, Jan. 2008.
- [13] LOCONSOLE, A., AND BÖRSTLER, J. An industrial case study on requirements volatility measures. Appendices: <http://www.cs.umu.se/research/software/apsec05/>. In *12th IEEE Asia Pacific Conference on Software Engineering* (Taipei, Taiwan, Dec. 2005), pp. 249–256.
- [14] MACDONELL, S. Metrics for database systems: An empirical study. In *IEEE METRICS* (1997), pp. 99–107.
- [15] NURMULIANI, N., ZOWGHI, D., AND FOWELL, S. Analysis of requirements volatility during software development life cycle. In *Proc. of the 2004 Australian Soft. Eng. Conf. (ASWEC'04)* (2004), p. 28.
- [16] PFAHL, D., AND LEBSANFT, K. Using simulation to analyse the impact of software requirement volatility on project performance. *Inf. & Soft. Tech.* 42, 14 (2000), 1001–1008.
- [17] RAYNUS, J. *Software process improvement with CMM*. Artech House, Inc., Norwood, MA, USA, 1999.
- [18] SCHNEIDER, G., AND WINTERS, J. *Applying use cases: a practical guide*. Addison-Wesley, Boston, MA, USA, 1998.
- [19] STARK, G., OMAN, P., SKILLICORN, A., AND AMEELE, A. An examination of the effects of requirements changes on software maintenance releases. *J. of Soft. Maint.* 11, 5 (1999), 293–309.
- [20] YOUNG, R. *The Requirements Engineering Handbook*. Artech House, 2004.