MEASURING REQUIREMENT EVOLUTION
A Case Study in the E-commerce Domain

Päivi Ovaska
Lappeenranta University of Technology, P.O. BOX 20, FIN-53851, Lappeenranta , Finland

Keywords: Requirement evolution, requirements creep, systems development prediction

Abstract: Although requirement evolution is a widely recognized phenomenon, there are only a few approaches for measuring it. These existing approaches are based on the assumption that all the requirements exist and can be seen in the requirement elicitation and analysis phases. They do not include provisions for the emergence during systems development of new requirements, which cannot be anticipated in the requirement elicitation and analysis phase. This paper shows how the concept of requirements creep is adopted for the measurement of emergent requirement evolution. We use a case study in the E-commerce domain to illustrate the use of this measure in the prediction of systems development. The findings of this study suggest that requirement evolution can be measured in a practical software project, and the findings demonstrate the strong influence of requirements creep on the systems development effort. The findings of our study encourage us to undertake further studies involving other organizations and projects.

1 INTRODUCTION

Changing requirements and requirement evolution are recognized as being a difficult and major source of risks in the systems development projects. Requirement evolution may lead to increasing costs, schedule overruns as well as to system evolution in software projects (Harker, Eason and Dobson, 1993; van Lamsweerde, 2000). There are many reasons for these changing requirements: stakeholders may change their minds about the functionality of the proposed system; new or modified requirements may emerge during the design, implementation and testing processes; analysts and designers may not understand the requirement etc.

Although requirement evolution is a widely recognized phenomenon, only a few approaches have been developed to quantitatively measure it. These approaches assume that all the requirements are gathered and documented in the beginning of the development of a systems and that the only changes and additions to these initial requirements are made during the systems development process. The objective of this paper is to study how requirement evolution can be measured in a situation, in which requirements are not known in the requirement specification stage but, rather, emerge during the course of the development of the system. This approach is based on the notion that the requirements for system development do no exist a priori but are, rather, socially constructed through interactions among the participants in system development (Davidson, 2002; Curtis, Krasner and Iscoe, 1988; Ovaska, 2003).

This paper presents the early results of an ongoing study into how requirement evolution should be measured. This information can be used to aid the prediction of system development in organizations through the collection of history information on completed projects. The collected history information can then be used to support the development of the prediction effort in future projects. The aim of this paper is to present ways of measuring requirement evolution using the concept of requirements creep, to demonstrate the use and importance of requirements creep in the case study as well as to gain an insight into how to further develop our approach.

The rest of the paper is structured as follows. Section 2 presents previous research into requirement evolution. Section 3 discusses a case study and an example of the use of requirements creep in the prediction of systems development. Section 4 presents the conclusions and discusses the research results along with topics for further studies.

Ovaska P. (2004).
MEASURING REQUIREMENT EVOLUTION - A Case Study in the E-commerce Domain.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 669-673
DOI: 10.5220/0002628806690673
Copyright © SciTePress

2 APPROACHES FOR REQUIREMENT EVOLUTION

There are at least two different approaches to managing requirement evolution. The requirement engineering approach (Wiegers, 1999; Kotonya and Sommerville, 1998; Jarke et. al, 1999) attempts to manage requirement evolution using process-oriented activities that try to eliminate the phenomena. The other approach (Jarke et. al, 1999, van Lamsweerde, 2000; Tomayko, 2002; Lehman, 1998; Lehman, Perry and Ramil, 1998) considers requirement evolution as a natural and inevitable feature of systems development instead of an issue that has to be solved or eliminated. Software lifecycle models and methodologies, such as prototyping (Kotonya and Sommerville, 1998) and, more recently, different kinds of agile methods (Cockburn, 2001), are examples of the latter approach to requirement evolution.

One way of understanding requirement evolution is to measure it quantitatively, as in (IEEE Std 982.1, 1988; IEEE Std 982.2, 1988; Andersson and Felici, 2002). These IEEE standards propose a Requirement Maturity Index (RMI); the RMI metric attempts to quantify the readiness of requirements. In (Andersson and Felici, 2001), the metric is extended by taking into account historical information on change, as a result of which a Historical Requirements Maturity Index (HRMI) is proposed. The above-mentioned approaches measure RMI or HRMI over all the software releases and quantify the readiness of requirements over time. These metrics are calculated on the basis of requirement specification, which is performed as a result of requirement elicitation and analysis.

A phenomenon called ‘requirements creep’ (Ryan et. al, 2001; Wiegers, 1999) takes into account new emerging requirements which do not necessarily exist in the requirement specification document but have emerged during the course of the development process. In (Wiegers, 1999), requirements creep refers to the ”difference between the requirements specification developed after the requirements procedure and the requirements at the time when the actual product is built”. In (Ryan et. al, 2001), requirements creep is referred as ”significant additions or modifications to the requirements of a software system throughout the lifecycle, resulting in extensions to and alteration of the software’s functionality and scope”.

3 CASE STUDY

3.1 Project Description

This study was carried out in the systems development department of an international ICT company. The project involved the development of an E-commerce mobile service platform. The system was intended to enable organizers or their sponsors to promote their products in different kinds of happenings, such as ice hockey and football games. The system was composed of two subsystems, a platform in which the services were running (Subsystem A) and a toolbox, which permitted addition, configuration and simulation services (Subsystem B). This toolbox was intended to run on a PC in the Windows environment and the platform in the UNIX environment.

The project employed the object-oriented approach for systems development and was implemented in 2001.The project was divided into the following phases on the basis of the company’s process model: pre-study, feasibility study, project execution and piloting and maintenance. The requirements were collected and analyzed during the pre-study and feasibility study phases. One of the first activities in the project execution phase was to design the software architecture, in which the system was divided into the respective modules and their interfaces. This was followed by the detailed design, implementation and integration and system testing phases. After the system testing showed the quality of the software to be acceptable, the product went onto the piloting and customer approval phases.

A process-oriented approach was taken to managing requirement evolution in the project. A requirement specification document was formulated, and the requirements were ‘frozen’ in this document upon completion of the requirement elicitation and analysis phases. After requirement freezing, requirements were managed through a strict requirement changing procedure. This requirement changing procedure was intended to be specified in each development project, but the spirit of the changing procedure was such that every single change to the requirements had to be analyzed and handled by the project steering group. Unfortunately, in most of the projects this did not work, as was the case in this E-commerce project. This project, involved an entire subsystem, Subsystem B, for which only four requirements were specified in the requirement specification document. Nevertheless, at the end of the development project, Subsystem B was 60 % of the size of the whole software application in terms of code size and the required development effort. The new requirements for Subsystem B emerged iteratively during the systems development through more than twenty prototypes made for Subsystem B (Ovaska, 2003).

3.2 Requirements Creep

We used the notion of requirements creep (Ryan et. al, 2001; Wiegers, 1999) and the conceptual modelling technique widely used in the database (Chen, 1976) and object-oriented approaches (Rumbaugh, 1990; Jacobson, 1992; Booch, 1993) to measure requirement evolution. In database modelling, conceptual modelling is called entity-relationships modelling and in the object-oriented approach it is known as object modelling. This conceptual modelling approach is a natural approach for object-oriented systems as it is an essential part of the object-oriented methodology.

We used two conceptual models of the system: an analysis model that described the domain concepts after requirement specification and an implementation model that described domain concepts after the implementation of the actual system. Based on these models, the number of different concepts between these models was calculated (requirements creep) (Table 1).

The analysis model was extracted from the project documentation and the implementation model was re-engineered from the source code after the completion of the project.

Table 1: The concepts found in the analysis and implementation models

Module	*Analysis model*	*Implementation model*
M1 (Subsys-tem A)	Alarm, Logging, Localisaction, Billing, Message, User, Service	Alarm, Logging, Localisaction, Billing, Message, User, Service
M2 (Subsys-tem A)	Messaging, Counting, Timing, Message, User	Messaging, Counting, Timing, Message, User, Simulation
M3 (Subsys-tem A)	Service control, Localization, Message, User, Service	Service control, Localization, Message, User, Service, Simulation
M4 (Subsys-tem A)	UI, Message, User, Service	UI, Message, User, Service
M5 (Subsys-tem B)	Service, Simulation	Service, Simulation, Messaging, Authentication, Localization, UI, Message, User

3.3 Prediction Model

The aim of this prediction model is to analyze the significance of the requirements creep (the correlation between the requirements creep and working hours) and demonstrate the use of the requirements creep.

We used a simple linear prediction model for the analysis, for which we required four other measurements (system properties) to characterize our system. We chose a widely used coupling (Xia, 1996), size (in LOC) (Henderson-Sellers, 1996), cohesion (Henderson-Sellers, 1996) and complexity/communication (Chidamber and Kemerer, 1994). The values of these system properties are shown in Table 2.

Table 2: The system properties for the case study

Module	*M1*	*M2*	*M3*	*M4*	*M5*
Working hours	107	300	304	378	1767
LOC	1011	2176	1970	2843	12986
Complexity/ communication	88	91	160	99	607
Coupling	12	20	37	18	54
Cohesion	78	85	93	84	100
Requirements creep	0	1	1	0	6

The model is based on the simple notion that the development effort for a system can be expressed as a linear function of the properties of each module and coefficient values:

$$C_n\left(m_{n,1}, m_{n,2}, \ldots, m_{n,p}\right) = a_1 m_{n,1} + a_2 m_{n,2} + \ldots + a_p m_{n,p} \quad . \quad (1)$$

In the formula, $m_{n,q}$ corresponds to the value of property q in module n and a_p is the linear coefficient that corresponds this property.

To analyze the correlation between the system properties and development effort, the problem turned out to be a function optimization problem, more precisely a non-negative least-square problem (Lawson and Hanson, 1995). The unknown factors of this model were the values of the coefficients. We used the MATLAB® Optimization Toolbox function lsqnonneg.m (Matlab, 2003) to solve the coefficient values.

3.4 Findings

Table 3 shows the values of the coefficients that were of some significance (value>0) in the model. The information in the table illustrates that requirements creep was the most significant measurement for the system (with the value 19). Coupling and Lines of Codes (LOC) also had some significance but much less than did requirements creep (with values 0.9 and 0.1).

Table 3: The values of the coefficients of the software assesment model

LOC	*Coupling*	*Requirements creep*
0.1234	0.9164	19.2901

The significance of LOC and coupling as cost factors are line with the literature (Briand, Daly and Wüst, 1999; Lionel et, al, 1998). Requirements creep has such a strong impact on the development effort that the other measurements are negligible.

4 DISCUSSION, CONCLUSIONS AND FUTURE WORK

This study focused on using the concept of requirements creep in measuring and introducing requirement evolution in a case study in the E-commerce domain. Requirements creep measures new emerging requirements which do not exist at the beginning of the development process but, rather, emerge through interactions between the participants in the development of the system.

The findings of this study suggest that requirement evolution can be measured in a practical software project and indicate the strong influence of requirements creep on the systems development effort.

It is not possible to generalize the results on the basis of one case study. More studies must be performed in other organizations and development environments in order to obtain more evidence on the applicability of our approach.

The main problem with measuring requirements creep was that the concepts in the application domain were slightly abstract and were not necessarily at the same level. Measuring these abstract concepts required a basic understanding of the concepts in the application (application knowledge) and definitions of these concepts during the project.

In the analysis, we used a linear prediction model that is too simple for real prediction purposes. In our prediction model, we assumed that the system development effort was a sum of the efforts of the modules. It is well known that small changes in the system size can have big effects on the development effort. There are prediction methods and models, such as case-based reasoning (Pedrycz, Peters and Ramanna, 1999) and neural networks (Venkatachalam, 1993), which take this non-linear nature of systems development into account.

The results of this study have encouraged us to analyse other industrial projects in order to collect more evidence on the usefulness of measuring requirements creep in other projects and domains. We will develop our linear prediction model towards a more non-linear approach in order for it to be scalable for real-life prediction purposes.

REFERENCES

Andersson, S., M. Felici, 2001. Requirements Evolution: From Process to Product Oriented Management. In *3rd International Conference on Product Focused Software Profess Improvement*, Springer-Verlag.

Andersson, S. M. Felici, 2002. Quantitative Aspects of Requirement Evolution. In *26th Annual International Conference on Computer Software and Applications Conference*, Oxford, England, IEEE Computer Society.

Booch, G., 1993. *Object-Oriented Analysis and Design with Applications*, Addison Wesley Pub Co, 2 nd editions.

Briand, L.,C., J. W. Daly, J. K. Wüst, 1999. A Unified Framework for Coupling Measurement in Object-Oriented Systems. In *IEEE Transactions on Software Engineering*, 25(1).

Chen, P., P., 1976. The entity-relationship model: toward a unified view of data. In *ACM Transactions on Database Systems*.

Chidamber, S.,R. and Chris F. Kemerer, 1994. A Metric Suite for Object Oriented Design. In *IEEE Transactions on Software Engineering*, Vol. 20, No. 6.

Cockburn, A., 2001. *Agile Software Development*. Addison-Wesley.

Curtis, B., H. Krasner, and N. Iscoe, 1988. A Field Study of the Software Design Process for Large Systems. In *Communications of the ACM 31*.

Davidson, E.J., 2002. Technology Frames and Framing: A Socio-Cognitive Investigation of Requirement Determination. In MIS Quarterly, Vol. 26, Issue 4.

Harker, S., K.Eason, and J. Dobson, 1993. The change and evolution of requirements as a challenge to the practice of software engineering. In IEEE International Symposium on Requirements Engineering, pages 266–272, IEEE Computer Society Press.

Henderson-Sellers, B.,1996. *Object-Oriented Metrics: Measures of complexity,* Prentice Hall, New Jersey.

IEEE Std 982.1, 1988. *IEEE Standard Dictionary of Measures to Produce Reliable Software*.

IEEE Std 982.2 , 1988. *IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software.*

Jacobson, I., 1992. *Object-Oriented Software Engineering*. Addison Wesley Pub Co, 1 st edition.

Jarke, C. Rolland, A. Sutcliffe, R. Dömges, 1999. *The nature of Requirements Engineering*. Aachen: Shaker Verlag GmbH.

Kotonya, G., I.Sommerville, 1998. Requirement Engineering, John Wiley & Sons, NY.

Lawson, C.L., R. J. Hanson, 1995. *Solving Least Squares Problems*. Society for Industrial & Applied Mathematics.

Lehman, M.,1998. Software's future: Managing evolution, In *IEEE Software*, Jan-Feb, pages 40–44.

Lehman, M., D. Perry, and J. Ramil, 1998. On evidence supporting the feast hypothesis and the laws of software evolution. In *Metrics '98,* Bethesda, Maryland.

MATLAB® the Language of Technical Computing, 2003. The MathWorks, Inc., [URL: http://www.mathworks.com/, Referred 20 Sep 2003].

Ovaska, P., 2003. On the Organizational Factors in Understanding of Information System Requirements. In *Scandinavian Conference on Information Systems* (IRIS26), Finland August 9-12.

Pedrycz, W., J.F. Peters, S. Ramanna, 1999. A Fuzzy Set Approach to Cost Estimation of Software Project. (Editor: Meng, M.), In *IEEE Canadian Conference on Electrical and Computer Engineering*.

Rumbaugh, J., 1990. *Object-Oriented Modeling and Design*. Prentice Hall.

Ryan A. R. Carter, I. Annie, I. Antón, A. Dagnino, L. Williams, 2001. Evolving Beyond Requirement Creep: A Risk-Based Evolutionary Prototyping.In *IEEE 5th International Symposium on Requirements Engineering*.

Tomayko, J.E, 2002.Engineering an unstable requirements using agile methods. In *International Workshop on Time Constraint Requirement Engineering*.

van Lamsweerde, A., 2000. Requirements engineering in the year 00: A research perspective. In International Conference on Software Engineering (ICSE'2000), pages 5–19, Limerick, Ireland.

Venkatachalam, A., R., 1993. Software Cost Estimation Using Artificial Neural Networks. In *International Joint Conference on Neural Networks*, October 25-29, Volume 1, Pages: 987 – 990.

Wiegers, K.E., 1999. Software Requirements, Microsoft Press.

Xia, F.,1996. Module Coupling: A Design Metric.In *Asia-Pacific Conference on Software Engineering.* Seoul, South Korea, 4-7 Dec. Pages: 44 – 54.