International Journal of Students Research in Technology & Management
Vol 1 (06), October – December 2013, ISSN 2321-2543, pg 619-623

# IMPROVING DEVELOPMENT TIMES BY REUSING FLEXIBLE HARDWARE AND SOFTWARE COMPONENT

**Bhaumik Paresh Mistry**
Department of Electronics and Telecommunications,
Rajiv Gandhi Institute of Technology, Mumbai, India.
Email: bhaumikpmistry@gmail.com

***Abstract***

***Product development time estimation is important for project management tasks. This study investigates the impact of requirements reuse on product development duration for different products in a similar domain. We propose an analytical tool to estimate the minimum time to be saved given the percentage of requirements reused from earlier projects. This paper presents basis for the formal reuse of hardware components as a strategy to reduce the task for verifying new hardware elements. Assuming the existence of a library of formally verified hardware components, we propose to make effective reuse of these existing elements when creating new ones. The concept of software component reuse is simple: the idea of building and using "software preferred parts." By building systems out of carefully designed, pre-tested components, one will save the cost of designing, writing and testing new code. The practice of reuse has not proven to be this simple however, and there are many misconceptions about how to implement and gain benefit from software reuse. One case from organization has been studied for software and system development projects, which consist of hardware and software components. The results of the case studies are compared with a study in the literature on product development time.***

**Keywords:** Hardware reuse, Software reuse, Griffin formula, Time development, Recycle.

## 1. INTRODUCTION

In most organizations, project deliverables are generally systems that combine hardware and software components. For system projects, which include hardware and software components, software is becoming an increasingly significant constituent [1][2] and project management is accordingly becoming more complex.

Engineers working in these projects discover most of the problems at the integration phase. Isolation of the source of these problems at this stage can take time and this may affect the project duration. According to [3], 50% of total time and cost of a project is spent for testing. So, minimizing possible system faults at earlier stages will minimize the test efforts. Since most defects are found in the integration and test phases, these phases are generally stressful for developers and testers since they are responsible for correcting the defects. Therefore, reuse of the components created during each phase of different projects has an important role in eliminating the defects in the product, correspondingly reducing the engineering effort in the project.

Reuse is the key to progress in any area. If we do not reuse previously development ideas and products then everything must be created from scratch and no progress can be made. In the development of software, we routinely reuse knowledge in the form of experience, processes in the form of methods, and products in the form of tools. Although the reuse of software components is less commonplace, there are certain well-known situations, such as in the case of mathematical or i/o routines, where the development and use of reusable components has been very successful.

The research objective in this area is to discover how to achieve a systematic way of reusing existing products, processes and knowledge to get the maximum cost benefit. Successes inn the reuse of software are due to both the characteristics of the reused item themselves and to the environment in which they are reused.

In practice, there are many technical, processes, economic and organizational issues to overcome. However, several companies find that reuse does work, and can payoff handsomely with high levels of reuse. There are significant corporate reuse programs at AT&T, HP, IBM, GTE, NEC, Toshiba, and others. They see major improvements in time-to-market, quality and costs [4].

To be successful, a reuse program requires much more than just reusable code modules and library technology. Careful consideration must be given to the way reusable work products are designed and packaged, to the architecture and framework in which reusable work products are combined, and to organizational and economic structures.

This study focuses on reducing some of the sources of defects hence the product development time via requirements reuse. The motivation behind the reuse of requirements items from previous projects is that these are already validated and accepted by end users in previous projects. In this study, one case study is presented to demonstrate the benefits of requirements reuse for a system project containing both hardware and software. In the literature, there are some studies on the reuse of some components and they have generally been performed for software code reuse. One of the contributions of the present study is to provide empirical results, which demonstrate the reduction of project duration due to requirements reuse for industrial products. Another contribution is to show that the product development time estimations proposed earlier in the literature for

manufacturing industries [5] can be applied to system and software projects. We also propose a modification to the current formulation to represent software-based projects more accurately [5].

## 2. RELATED WORK

This section reviews the literature on Software Product Line (SPL) and product development time.

### *A. Software Product Line*

When reuse is applied in all stages of the product development cycle, this corresponds to the product line concept. The aim of the product line concept is to enhance the quality of the product and decrease the engineering cost. In the literature, there are many studies, which evaluate the establishment of software and system product lines in organizations [6][7][8][9].

The accepted SPL approach is based on two-life cycles, which are Domain Engineering (DE) and Application Engineering (AE). SPL mostly focuses on the DE activities [6], which create a common infrastructure for the related domain. On the other hand, AE is active when there is a new project. To produce a product, the required assets are selected from the common assets created by DE. For the remaining part of the product, new assets are created from scratch. Although SPL applies to all phases of projects life cycle, this study covers only the requirements definition phase. Thus, in the following sections, requirements definition phase, which is a part of SPL, is studied.

### *B. Product Development Time*

While preparing project proposals, little information is available concerning the development details. Before starting the development, it is important to have information about development time to estimate the cost of the project used for proposals.

In the literature, there is some information about the methods for estimating the duration for product development. Griffin [13] classifies the metrics, which affect the development time in four groups as;

- Changes during the product generation,
- The complexity of product,
- Whether a formal process is used in the organization, and
- Whether a cross functional team is used.

Changes in requirements have a considerable effect on the development workload ([10][14][15]). Therefore, it is important to define the requirements accurately to ensure minimum change during the development stage. The number of main functions the product performs gives the complexity of the product [13]. If organizations do not have formal development processes, the development time is higher compared to those with formal development processes. A study by Olson et al. [11] emphasizes that the use of a cross functional team is also an important parameter in increasing project performance. Griffin [12][13] defines Development Time (DT) and Concept To Customer (CTC) as two separate parameters. DT begins from design up to the introduction to the customer and CTC begins with concept development and continues to specification definition until the introduction to the customer. Requirements engineering activities are covered within CTC. If DT is subtracted from CTC this will give the time spent for requirements engineering activities. DT and CTC formulations proposed for the manufacturing industries are given below [13].

$$DT=\alpha+\beta_{1DT} * PC+\beta_{2DT} * NN+ \beta_{3DT} *(PC * FP)+\beta_{4DT} *(NN * XFT)+\varepsilon_{DT} \quad (1)$$

$$CTC=\alpha+\beta_{1CTC} * PC+\beta_{2CTC} * NN+\beta_{3CTC} *(PC * FP)+\beta_{4CTC} *(NN * XFT)+\varepsilon_{CTC} \quad (2)$$

Where $\alpha$ is the cycle time constant, PC and NN are product complexity and product newness/uncertainty, respectively. If NN value increases, the change probability on the product during the development also increases. FP and XFT show whether formal processes and cross functional teams are used, respectively. If formal development processes are not used, then FP=0. $\varepsilon$ is the error term. The unit of $\beta_1$ and $\beta_3$ are the months/function designed in the product. The unit of $\beta_2$ and $\beta_4$ are the months/percentage of change in the product. The estimation of the coefficients $\alpha$ and $\beta$, based on data collected from many companies, is given in Table I [14].

## 3. FACTS ABOUT REUSE

Because reuse is such a simple concept, we have found many misconceptions about it in discussions with software developers and software engineering researchers. The panelists are expected to explain what they view as the most important facts and myths about reuse. I've explained a few of my own below.

### *The software development process*

Effective reuse is not a simple addition to existing software development processes. The systematic application of software reuse will dramatically change the software process [3]. Two of the most important changes are:

Changes in the way software entities will organize themselves to produce and consume reusable software work products. Effective software reuse needs a clear division among the roles of producers of reusable workproducts and consumers who use them to build applications. Changes in management and support

structures are needed to support the new roles. Changes in the way developers are evaluated and rewarded may be required to reinforce the new roles.

Changes will be made in software development models, methods, processes and technologies to take advantage of reuse. Applications will no longer be designed independently, one at a time. Instead, conscious effort will be made (Domain Engineering) to design reusable assets for a range of expected applications and new applications will be designed to take maximum advantage of the available software preferred parts.

***Reuse payback***

When a reuse program is started, the organization needs to have appropriate expectations on when a return on the investment will be realized. Reuse has proven to be a long-term investment showing benefits over a series of projects often spanning several years [4]. When a short-term payback is expected, disappointing early results can cause a program to be canceled before it has a chance to pay off. Benefits of a reuse program are more than cost savings on projects. Benefits of earlier time to market and of decreased long-term maintenance costs must also be taken into account when evaluating a reuse program.

Since reuse changes roles and organizations and may require new organizational structures, funding the production, use, and maintenance of components can be a thorny issue in today's business environment. The reuse producers may not be attached to a particular profit producing project and can be viewed as a cost center. This leaves them vulnerable to cuts when budget adjustments must be made.

## 4. CHARACTERISTICS OF CASE STUDY

To analyze the product development time using the requirements reuse approach, it was intended to gather data from different companies and the requirements engineering phase would be analyzed for three different organizations. Data is collected with a joint effort of the project technical managers and the authors.

The project related data are derived from the responses to the questions given below, following the methods as described:

***Question 1*:** Are there any similar products that can be in the same domain or are derivative products in the company?

*Method Used for Answering Question 1:* Discussions with different project technical managers from different companies are performed and the details of the projects are evaluated.

***Question 2:*** Is there recorded data for the number of requirements for each project in the same domain?

*Method Used for Answering Question 2:* The System Requirements Documents for each project are used to obtain the necessary data.

***Question 3:*** Is there recorded data for reused requirements?

*Method Used for Answering Question 3:* If the metrics are kept systematically, data is retrieved from organization database. If they are not kept in an organizational database, technical personnel involved in the projects derive reused requirements from the system requirements documents.

***Question 4:*** Is there duration data for requirement definition phases?

*Method Used for Answering Question 4:* The enterprise resource planning systems of the companies are used to extract this data.

***Question 5:*** What is the complexity level of the product to be studied?

*Method Used for Answering Question 5:* the main functions of the products are defined to obtain the complexity level of the product with the help of technical managers of the projects. If there is critical technology to be developed within the scope of the product, this factor is also added to the complexity.

## 5. CASE STUDY

Project A1 and Project A2 in the same domain from Company A were analyzed. Some of the requirements of Project A1 have been reused in Project A2. Table II shows the number of requirements in Project A2. Project A2 had 183 requirements in total. 104 requirements of those were reused from Project A1. Remaining 79 requirements were created from scratch for Project A2. The realized duration for requirements engineering (RE) activities for both projects is given in Table III. Change probability of 104 reused requirements was very low in Project A2, because they were tested, and approved by the customer previously. This implies that;

57% of total requirements (104 requirements) for Project A2 were almost fixed.

43% of total requirements (79 new requirements) could still be changed in Project A2. By reusing the requirements; change probability in the product (NN) is minimized. While NN varies between 0% and 100%, by reusing the requirements it can be decreased in the range of 0% to 43% for Project A2. By using Griffin's CTC formulation (2), this situation indicates that for all possible changes in the requirements, the organizations would require an additional 16 months ($\beta 2C\ T\ C * N\ N = 0.16 * 100\%$) if requirements were not reused. On the other hand, the organization would only require an additional 6.88 months (0.16*43%) maximum when requirements were reused. So, change effect is reduced by 9.12 months for Case Study.

To estimate the time spent for RE activities, the calculations of DT and CTC given in Appendix-A for 100% and 43% cases are performed using (1) and (2). 100% indicates that product requirements/features were totally

new. 43% indicates the amount of new requirements, and is taken as the change probability of the requirements. The complexity level of the product developed in the scope of Project A2 was taken as 6 based on the number of main functions the product has.

Even if the maximum change (43%) occurs in the requirements, there would be at least 22% decrease (from 18 months to 14.01 months) in the duration of RE activities. If the change in the requirements were less than 43% change, the improvement would be expected to be greater than 22%.

When this result is compared with the actual results of Case Study in Table III, the decrease in the Project A2 shows agreement with these calculations. Griffin's formulation predicts at least 22% reduction in duration; likewise a reduction of 37% (more than 22%) was obtained. Thus, formulations proposed by Griffin for the estimation of project duration apply for the system, which involves both hardware and software.

## 6. APPLICATIONS

Electrical and electronic waste (e-waste) is currently the largest growing waste stream. It is hazardous, complex and expensive to treat in an environmentally sound manner, and there is a general lack of legislation or enforcement surrounding it and proper reuse of the hardware. Which can reduce the e-waste and its effects on environment.

If the e-waste in developed countries that are sent for recycling, 80 per cent ends up being shipped (often illegally) to developing countries such as China, India, Ghana and Nigeria for recycling.

**Long-term effects on human health and the environment**

It is evident from several studies in China that the rudimentary recycling techniques coupled with the amounts of e-waste processed have already resulted in adverse environmental and human health impacts, including contaminated soil and surface water (Zhao et al., 2010; Wang et al., 2011; Frazzoli, Orisakwe, Dragone & Mantovani, 2010; Tsydenova & Bengtsson, 2011). Health problems have been reported in the last few years, including diseases and problems related to the skin, stomach, respiratory tract and other organs (Nordbrand, 2009).

## 7. CONCLUSION AND FUTURE WORK.

The focus of this study is the reuse of requirements for different products in similar domains. The effects of requirement reuse for two different product types, one consisting of hardware and software, and the other purely software, have been investigated. For this investigation one case study has been performed and its result have been compared with a theoretical study. It is very likely that different projects in the same domain have many common requirements and if these requirements were maintained and shared in a common database that employees could access, systems engineers would choose to use these requirements in different projects. In the context of such an opportunity, the product would be developed within a common understanding of the requirements. Besides, availability of applicable product development time estimations has an importance on the project management tasks. By using an applicable estimation model for industrial products, it should be possible to make project budget and resource allocation with minimum error. It might be difficult to work with minimum error at the beginning of the project with less information about the development details. From the case studies it is concluded that the proposed method can be applied to system projects with hardware and software. The reuse of components can improve the day-by-day increment in e-waste; this can change the total effect of the waste product on environment. The proposal of the reuse of hardware as well as software is beneficial. However, it does not yield the same results with real-life software projects. This is because software requirements may change more easily when compared to hardware requirements. So, Griffin's formulation is revised for the software products and proposed formulation has more effect for the change probability of the product.

This study showed the effects of requirements reuse on project duration based on empirical results of the case studies in which the data are collected from industry. This study covers only the requirements analysis phase. In terms of the future work, investigating reuse in the other phases of the project life cycle can further reduce the duration. Further changes can be done, to reduce the maximum damage done by the e-waste to environment.

**Table 1. Coefficients used in the DT and CTC formulation.**

| | A | $\beta_1$ | $\beta_2$ | $\beta_3$ | $B_4$ |
|---|---|---|---|---|---|
| DT | 8.4 | 4.2 | 0.09 | -1.9 | -0.09 |
| CTC | 10.4 | 3.7 | 0.16 | 0.1 | -0.16 |

**Table 2. Requirement used in project A2** .

| | Total # of Req. in Project A2 | Req. of Project A1 Reused in Project A2 |
|---|---|---|
| Total | 183 | 104 (57% of Project A2) |

**Table 3. Duration expended in project a1 and project A2.**

| | Project A1 | Project A2 | Possible Impact of Reuse |
|---|---|---|---|
| Total Duration (months) | 8 | 5 | 37 % decrease in duration |

**Calculations for Case Study**

$CT\ C_{100} = 10.4+3.7*6+0.16*100\%+0.1*6 = 49.2$ months

$CTC_{43} = 10.4+3.7*6+0.16*43\%+0.1*6 = 40.08$ months

$DT_{100} = 8.4+4.2*6+0.09*100\%-1.9*6 = 31.2$ months

$DT_{43} = 8.4+4.2*6+0.09*43\%-1.9*6 = 26.07$ months

The time spent for requirements engineering is:

$CTC_{100} - DT_{100} = 49.2 - 31.2 = 18$ months

$CTC_{43} - DT_{43} = 40.08 - 26.07 = 14.01$ months

**REFERENCES**

1. Matthew R. Kennedy and David A. Umphress, "An Agile Systems Engineering Process - The Missing Link", CrossTalk, The Journal of Defense Software Engineering", May-June 2011.
2. Richard Turner, "Toward Agile Systems Engineering Processes", Cross Talk, The Journal of Defense Software Engineering, April 2007.
3. Shaojie Guo, Weiqin Tong, Juan Zhang, and Zongheng Li, "An Appli- cation of Ontology to Test Case Reuse", International Conference on Mechatronic Science, Electric Engineering and Computer, pp. 775-778, 2011.
4. Martin L. Griss, John Favaro, and Paul Walton. Managerial and Organizational Issues - Starting and Running a Software Reuse Program, chapter 3, pages 51-78. Ellis Horwood, Chichester, GB, 1993.
5. Abbie Griffin, "The Effect of Project and Process Characteristics on Product Development Cycle Time", Journal of Marketing Research, Vol. 34, No. 1, pp. 24-35, February 1997.
6. Timo Kkl, "Standards Initiatives for Software Product Line Engineering and Management within the International Organization for Standard- ization", Proceedings of the 43rd Hawaii International Conference on System Sciences, pp. 1-10, 2010.
7. Sholom Cohen, "Guidelines for Developing a Product Line Concept of Operations", Technical Report. CMU/SEI-99-TR-008, August 1999.
8. Frank Dordowsky, Richard Bridges, and Holger Tschpe, "Implementing a Software Product Line for a complex Avionics System", 15th Inter- national Software Product Line Conference, pp. 241-250, 2011.
9. Kentaro Yoshimura, Jun Shimabukuro, Takatoshi Ohara, Chikashi Okamoto, Yoshitaka Atarashi, Shinobu Koizumi, Shigeru Watanabe, and Kazumi Funakoshi, "Key Activities for Introducing Software Product Lines into Multiple Divisions: Experience at Hitachi", 15th International Software Product Line Conference, pp. 261-266, 2011.
10. Andy J. Nolan, Silvia Abraho, Paul C. Clements, and Andy Pickard, "Requirements Uncertainty in a Software Product Line", 15th Interna- tional Software Product Line Conference, pp. 223-231, 2011.
11. Eric M. Olson, Orville C. Walker, Robert W. Ruekert, and Joseph M. Bonner, "Patterns of Cooperation During New Product Development Among Marketing, Operations and R&D: Implications for Project Per-formance", Journal of Product Innovation Management, Vol 18, Issue 4. pp. 258-271, July 2001.
12. Abbie Griffin, "Metrics for Measuring Product Development Cycle Time", Journal of Product Innovation Management, Vol. 10, No. 2, pp. 112-125, March 1993.
13. Abbie Griffin, "The Effect of Project and Process Characteristics on Product Development Cycle Time", Journal of Marketing Research, Vol. 34, No. 1, pp. 24-35, February 1997.
14. He-Biao Yang, Zhi-Hong Liu, and Zheng-Hua Ma, "An Algorithm for Evaluating Impact of Requirement Change", Journal of Information and Computing Science, Vol. 2, No. 1, 2007.
15. Susan Ferreira, James Collofello, Dan Shunk, and Gerald Mackulak, "Understanding the Effects of Requirements Volatility in Software Engineering by Using Analytical Modeling and Software Process Simulation", Journal of Systems and Software, Vol. 82, Issue 10, pp. 1568- 1577, October 2009