International Journal of Computing and Business Research (IJCBR)

ISSN (Online) : 2229-6166

Volume 3 Issue 3 September 2012

COMPONENT BASED SOFTWARE REUSABILITY: A CASE STUDY

Jatinder Khetarpal
Govt. P.G. College, Sector-14, Karnal
jatinderkhetarpal@yahoo.co.in

Abstract
Software reuse is one of the most economical approaches to enhance programmer productivity and software quality. Although the use of software development methods and tools is increasing and the expenditure on software development is reducing, still there is scope for further improvements in the production of new software. An effort has been made to adopt development models to provide an adequate framework to support the concept of reuse and its planned application to the software development.

Keywords: Software reuse library, SDLC, Domain Engineering, Frame working

1.0 Introduction

Software reuse is the process of implementing or updating software systems using existing software assets. It is the process of creating software systems from predefined software components designed for reuse. It has long been recognized by computer scientists that reuse is a powerful technology that can deliver tremendous benefits. People's perspective may be different in visualizing software reuse ranging from reuse as a technique for improving productivity and quality to view it more as a collection of available components. When reuse is mentioned, one often thinks only of code reuse. This is perhaps one of the lesser productive forms of reuse. Categorically there are rewards to be gained in considering inheritance, template, component, framework, pattern and domain component reuse but worth reuse articulates are previously created cases, standards documents, models, procedures and guidelines. Reuse eventually saves time and money, and will ultimately lead to a more stable and reliable product. Levels of reuse are Application system reuse, Subsystem reuse and Function reuse .

2.0 Merits and demerits of Reuse

Implemented properly reuse can aid in achieving some or all of the following goals:

- Increased Reliability: components already exercised in working systems
- Efficiency: Reduces time spent designing or coding
- Debugging: Reused design/code is often tested design/code.
- Profit : Reuse can lead to a market for component software; Real-world examples : ActiveX components, Hypercard stacks, Java packages, even software tools
- Reduced Process Risk: less uncertainty in development costs
- Effective Use of Specialists :reuse components instead of people
- Standards Compliance : embed standards in reusable components
- Accelerated Development : avoid custom development and speed up delivery

Demerits of Reusability may include Increased maintenance costs, Lack of tool support, Pervasiveness of the "not invented here" syndrome, Need to create and maintain a component library and Finding and adapting reusable components etc.

3.0 Literature Survey:

Many software organizations around the world have reported successful reuse programs such as IBM, Hewlett-Packard, Hitachi and many others [1 Griss93].
Reuse is an umbrella concept, encompassing a variety of approaches and situations[2 Morisio02]. Mili et al. define reusability as a combination of two characteristics [3 Mili02]: 1. Usefulness, which is the extent to which an asset is often needed. 2. Usability, which is the extent to which an asset is packaged for reuse. Frakes et al. have investigated 16 questions about software reuse using a survey in 29 organizations in 1991-1992 [4 Frakes95]. Jacobson et al.'s book describes the reuse-driven software engineering business to manage business, architecture, process and organization for large-scale software reuse [5 Jacobson1997]. There is extensive literature on reuse, from books [e.g., 6 Bosch 2000; 7 Clements and Northrop 2001; 5 Jacobson et al. 1997; , to the reports of the Software Engineering Institute [8 SEI 2005] and to papers in various technical journals and conferences. [Mohagheghi and Conradi 2007] This product shares reusable components with a second product in a product family, and the reuse rate is around 60%. Here is an overview of a few reported qualitative findings:
Reuse allows a company to use personnel more effectively because it leverages expertise (Lim 1994). More experienced personnel can be assigned to develop the reusable assets. Selby (2005) reported that larger projects reuse more with modification than smaller ones since scale may motivate reuse. Mohagheghi et al. (2004) reported that a reusable architecture leads to clearer abstraction of components. Reuse and standardization of software architecture and processes allowed also easier transfer of development in the conditions of organizational changes (Mohagheghi and Conradi 2007).

4.0 Technologies That Support Reuse (Tools)

Technologies that support reuse are the tools used to develop, find, understand, improve, integrate, and test reusable components. The problem is lack of technology that would support reuse. Some technological problems include lack of (1) techniques for building and maintaining component libraries; (2) tools for promoting reuse; and (3) methodologies supporting modification and use of reusable components, especially for maintaining software. Once the software development organization has been assessed and a plan of software reuse defined, a set of activities that support reusable assets based methodology installation should be conducted.

4.1 Case Tools

International Journal of Computing and Business Research (IJCBR)
ISSN (Online) : 2229-6166
Volume 3 Issue 3 September 2012

Software engineering methods provide the technical "how to" for building software. Software engineering tools provide automated or semi-automated support for methods. A tool is an artifact that one uses to make task easier (or possible). It has no intrinsic value except what it enables one to build. The term "software environment" means a collection of tools, practices, and working conditions supporting software engineering efforts [Lim 1994]. When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called Computer-Aided Software Engineering (CASE).

Table 1. Comparison of Reuse Rate with Defect Density

Subsystem	Software Size	Library Units	Reuse Percentage	Defect Density
1	27	0.338	0.44	6.2
2	5	0.718	0.941	0.6
3	6	0.923	0.741	0.4
4	3	0.512	0.09	8.0

5.0 Best Practices

Applying a certain paradigm or technology requires a change in attitude and needs to allow sufficient time for the team members to mature in their roles. Every developer must have the time to reach some level of maturated experience. Most developers need to be able to use their existing assets to build new ones and to learn something useful from their past experience. Reuse should be applied as a "first principle." That is, reusable products should always be considered as the basis for work before creating new products. The reality is that, a development team will likely fail if it sets out to craft a framework from scratch. The most successful framework seems to come from harvesting results and generalizing experience from earlier successful systems in the same domain. A smaller number of developers should be skilled in crafting new classes.

6.0 Proposed Plan :

The model will be based on Component based Development model. It has been widely accepted in both academia and industry for building software with reusable components. In this model the process starts by requirements analysis followed by system specification and system design as usual. It starts with the domain-engineering phase in parallel to requirements analysis phase. This is followed by frame working during which the interrelationships between components establish. Building system from frameworks is faster and easier than starting from scratch. The proposed model will follow a top-down or bottom up strategy for software development, taking into consideration the knowledge that a software engineer has about the application domain. In the proposed model aim is to cover all likely phases of large software development and ensures the important aspect of software reusability along its phases.

In the proposed model the aspect of development for reuse and development with reuse, will be taken into consideration. The proposed model supports the “development for reuse” through component archiving and the “development with reuse” through component assembly. With this the system development will become easy in the process. During the last phase the software engineer can come out with the components that can be reused in the future applications, and put them in the component pool, implementing the concept of “development for reuse”.
It will focus on the idea of component reusability. It can be evaluated by reuse cost, reuse efficiency as parameters. With proper implementation the other benefits that can be highlighted are increased productivity, shortened time to market with reduced risk, improving software Quality, reducing cost, improving functionality and performance.

The phases of the proposed model are

Domain Engineering, Frame working, System analysis, System design, Implementation [Component Selection, adaptation, testing], Integration, System testing, Deployment [Component Archiving], Maintenance.

The new model will address the issues of developing a new software system with preexisting component, thus it has great applicability in component based software development. According to this model, the appropriate components will be selected and integrated in the system. During the selection of components it is not necessary to get the same component that is exactly matching the requirements, sometimes one has to modify the requirements, so that the selected component can be used.

An effort will be made to implement the proposed model taking different case studies and using them in different languages. All efforts will be made to make the process bias free. The language selection is dependent upon organization and is mostly the choice of the programmer, still different languages will be used to make the study general in nature.

Metrics to be used will be: Reuse Cost Avoidance, Return on investment, and Reusability Factor

7.0 Conclusion

All different software development models have their own advantages and disadvantages. Nevertheless, in the contemporary commercial software envelopment world, the fusion of all these methodologies is incorporated. Timing is very crucial in software development. If a delay happens in the development phase, the market could be taken over by the competitor. Also if a fault filled product is launched in a short period of time, it may affect the reputation of the company. So, there should be a tradeoff between the development time and the quality of the product. Customers don't expect a bug free product but they expect a user-friendly product. The idea of being able to classify parts of a software system as potentially reusable is a powerful concept. It is a time and cost saving formula and indeed more time can be spend on the specific aspects of the software than general aspects of the software that might be needed for specific application. The development of a component should therefore be with generality with reuse and reengineering in mind placing perhaps less emphasis on satisfying the specific needs of an application that is being developed.

References:

1. Griss M.L., "Software Reuse: From Library to Factory". IBM Systems Journal, ed. 4, V. 32, pp. 548-566, November-December 1993. [Griss93].
2. Morisio M., Ezran, M., Tully, C. "Success and Failures in Software Reuse." IEEE Trans. Software Engineering, ed. 4, V. 28, pp. 340-357, April 2002. [Morisio02].
3. Mili H., Mili A., Yacoub S., Addy E., "Reuse-based Software Engineering. Techniques, Organizations, and Controls." John-Wiley & Sons, 2002. [Mili et al.2002].
4. Frakes W.B., Fox C.J. "Sixteen Questions about Software Reuse". Communications of the ACM, ed. 6, V. 38, pp. 75-87, June 1995. [Frakes95].
5. ITEA project, "ROBOCOP-Robust Open Component Based Software Architecture for Configurable Devices Project, http://www.hitech-projects.com/euprojects/Robocop.Jacobson I., Griss M., Jonsson P., "Software Reuse: Architecture, Process and Organization for Business Success", Addison Wesley, 1997, ISBN 0201924765. 1997. [Jaccobson, 1997].
6. Bosch J., "Design and Use of Software Architecture: Adpoting and Evolving a Product-Line Approach". Addison-Wesley. 2000. [Bosch 2000]
7. Clements P., Northrop L.M., "Software Product Lines: Practices and Patterns". Addison-Wesley. 2001. [Clements and Northrop 2001].
8. SEI 2005. http://www.sei.cmu.edu/, [SEI 2005].
9. Mohagheghi P. and Conradi R. "Quality, productivity and economic benefits of software reuse: a review of industrial studies." To appear in the *Empirical Software Engineering Journal*. 2007. [Mohagheghi and Conradi 2007]
10. Lim W. C., "Effect of reuse on quality, productivity and economics". IEEE Software, ed.5,V. ,pp. 23-30. 1994. Best paper award. [Lim 1994].
11. Selby W., "Enabling reuse-based software development of large-scale systems." IEEE Trans. Software Engineering, ed. 6,V. 31,pp. 495-510. 2005 [Selby 2005].
12. Clements P , Northrop L.M., "Software Product Lines. Addison-Wesley" .2002
13. Jalote P., "Software Engineering", 2004. [Jalote 2004].
14. Ashok K., Naveeta A. " A New Component Based Software Development Model" Published in International Journal of Ultra Scientist Of Physical Sciences". June 2007 Issue.