Towards the Measure of Reused Component Cost

N Md Jubair Basha
Assistant Professor, IT Department,
Muffakham Jah College of Engineering & Technology,
Hyderabad, INDIA.
jubairbasha@mjcollege.ac.in

Dr. M.Chandra Mohan
Associate Professor, CSE Department,
JNTUH College of Engineering,
Hyderabad, INDIA.
c_miryala@yahoo.com

Abstract— **Recent practices of software industry demands development of a software within time and budget which is highly productive. The demand of component based development and software reuse is increasing day by day. The issues related to the estimation of cost reuse measures are still challenging. This paper presents the cost for the reused components and proposes reuse measures for the family of applications with the quantized values. By analyzing these cost measures, the budget and effort in the development can be reduced. The results are estimated from the HR Portal domain specific application.**

Keywords- components; estimantion; software reuse, domain engineering

I. Introduction

Reuse level measures are used to develop the models for estimating the longer term cost and time savings due to reuse. This helps in motivating a move to a reuse business and also to make appropriate investment decisions and trade-offs, related directly to the overall business goals. In order to achieve these goals, several models are used to estimate overall quality, cost and time measures in terms of specific process and application and component system measures such as reuse levels, component reuse levels and complexities are needed. In this paper, several quantitative measures are presented for reused components to achieve the cost and measures for families of applications are presented.

The remaining part of this paper is organized as follows: Section II describes the related work for reuse metrics. The Section III consists of Software Reuse and Domain engineering process. The HR Portal application is realized with its components and their invocations in Section –IV. Section –V describes the estimation of reuse measures for the components and the overall application and for the family of applications has estimated.

II. Related Work

Reusability metrics defines an approach to measure reusable components. Several reusability metrics have been proposed in literature which has less emphasis on quantitative metrics. In [10] reusability metrics are based upon four characteristics viz. Self descriptiveness, modularity, portability and platform independence. However their weights are assumed based on assumed value which is qualitative in nature. In [11] a subset of reusability metrics are proposed. Though this approach is more efficient than non-automatable techniques, however the goal is to reuse the components interfaces only. This approach lacks the reuse measures at the design level. Zhongjie Wang et.al [12], proposed that the deficiencies of the components which are not suitable for reuse has to be redesigned. However no such approach for the overall system is presented. [15] proposed only measure for the victim components but for the family of applications is not presented.

III. SOFTWARE REUSE

Software Reuse is the use of available software or to build new software from software knowledge. Reusable assets can be either reusable software or software knowledge. Reusability is a property of a software asset that indicates it's probability of reuse [1]. Software Reuse means the process that use "designed software for reuse" again and again [2]. By software reusing, we can manage complexity of software development, increase product quality and makes faster production in the organization. Recently, design reuse has become popular with (object-oriented) class libraries, application frameworks, design patterns and along with the source code [3]. Jianli et al. proposed two complementary methods for reusing existing components. Among them one allows component evolution itself, which is achieved with binary class level inheritance across component modules.
The other is by defined semantic entity so that they can be assembled at compile time or bind at runtime. Although component containment still is the main reuse model that leads to contribute the software product lines development [4]. Regarding the components much information has to be collected, maintained and processed for the retrieval of the components. Maurizio has proposed a methodology to automatically build a software catalogue the tools for archiving and retrieval of information are presented [5].

Software Reuse can be broadly divided into two categories viz. product reuse and process reuse. The product reuse includes the reuse of a software component and by producing a new component as a result of module integration and construction. The process reuse represents the reuse of legacy component from repository. These components may be either directly reused or may need a minor modification. The modified software component can be archived by versioning these components. The components may be classified and selected depending on the required domain. [6].

3.1 Domain Engineering

Software Reuse can be improved by identifying objects and operations for a class of similar systems, i.e. for a particular domain. In the context of software engineering domains are application areas [7].

There are various definitions of what a *domain* is. Czarnecki's defines [8]:" an area of knowledge scoped to maximize the satisfaction of the requirements of stakeholders, which includes concepts and terminology understood by practitioners in the area and the knowledge of how to build (part of) systems in the area".

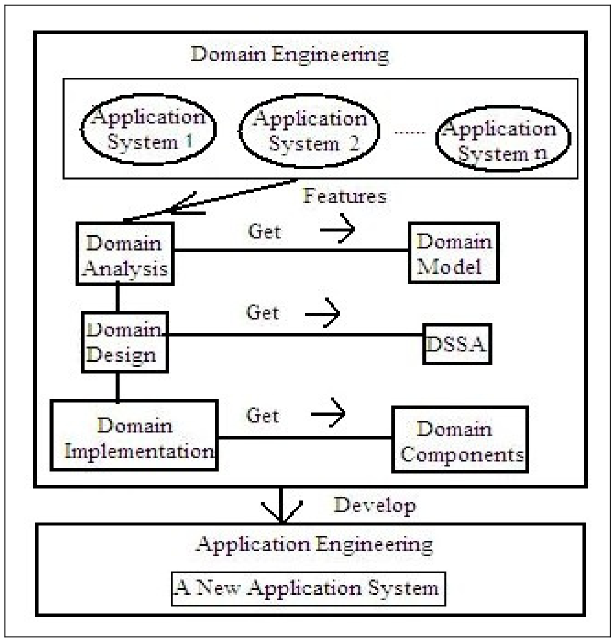


Figure 1 Process of Domain Engineering

Domain Engineering is a process in which the reusable component is developed and organized and in which the architecture meeting requirements of the domain is designed [9].Domain Engineering can be defined by identifying the candidate domains and performing domain analysis and domain implementation which includes both application engineering and component engineering. Domain Analysis is a continuing process of creating and maintaining the reuse infrastructure in a certain domain. The main objective of domain analysis is to make the whole information readily available. The relevant components (if available) has to be extracted from the repository rather than building the new components from the scratch for a particular domain.

Domain Analysis mainly focuses on reusability of analysis and design, but not code. This can be achieved by building common architectures, generic models or specialized languages that additionally improve the software development process in the specific problem area of the domain. A vertical domain is a specific class of systems. A horizontal domain contains general software parts being used across multiple vertical domains. Mathematical functions libraries container classes and UNIX tools are the examples of horizontal reuse. The purpose of domain engineering is to identify objects and operations of a class in a particular problem domain [7].

In the process of domain analysis, each component identified can be categorized as follows.

• General-purpose components: These components can be used in various applications of different domains (horizontal reuse).

• Domain-specific components: They are more specific and can be used in various applications of one domain (vertical reuse).

• Product-specific components: They are very specific and custom-built for a certain application, they are not reusable or only useful to a small extent.

IV. HR Portal Application

The system is designed in such a way that the client can interact with the web tier and business tier and can connect to the Data Access Object(DAO) component. The web-tier component consists of the JSP's and Servlets.The Business tier consists of the EJB's.The DAO's consists of the classes with its objects communicating to the database.The web-tier components are HttpServlet, HRProcessServlet, Login Servlet, InterviewResultServlet and RegistrationServlet classes.The Business-tier components are EmployeeBean, InterviewResultsBean, HRProcessBean are the three stateless bean classes.The DAO components are BaseDAO, EmployeeDAO, InterviewDAO, HRDAO, ProcessDAO classes.

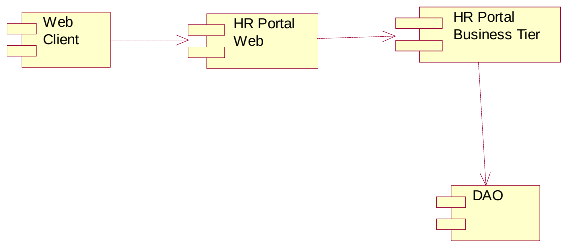


Figure 2: Components of HR Portal Application

Most of the reuse driven approaches often maintains a repository of reusable components. However an approach is needed to identify those components which are reused or might have been used very more. Such components are known as Non-Victim components.

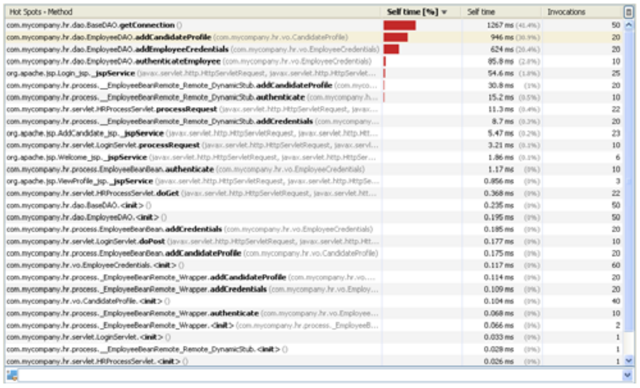
At any particular point of time, if the designer wants to know about which part of the system is not effectively reused then a lookup is to be performed on the component management relation. A Central repository maintains a table for managing a component reuse. This table contains two fields. One specifies the name of the component and the count specifies the number of times the component was reused by several systems.

Table 1. specifies the list of HR portal system components which were reused by several applications. If any component is not used frequently, they are termed as victim components. As Businesstier component was used only 10 times, it can be a candidate of victim component. The victim component has to be reconfigured by dividing it into several parts to increase the reusability count in future.

Table 1. Component Management Relation

Component	**Count of Reuse**
DAO	36
Web tier	10
Business tier	24

The count of reuse is achieved through by deploying the HR Portal application on to the Net Beans IDE. By Netbeans Profile, it facilitates about the how many times a component is invoked. In that application, how many times the components are invoked are presented. Considering the count which is showed in the Figure 2 the reuse cost measures can be estimated.



Hot Spots - Method	Self time [%]	Self time	Invocations
com.mycompany.hr.dao.BaseDAO.**getConnection** ()		1267 ms (41.4%)	50
com.mycompany.hr.dao.EmployeeDAO.**addCandidateProfile** (com.mycompany.hr.vo.CandidateProfile)		946 ms (30.9%)	20
com.mycompany.hr.dao.EmployeeDAO.**addEmployeeCredentials** (com.mycompany.hr.vo.EmployeeCredentials)		624 ms (20.4%)	20
com.mycompany.hr.dao.EmployeeDAO.**authenticateEmployee** (com.mycompany.hr.vo.EmployeeCredentials)		85.8 ms (2.8%)	10
org.apache.jsp.Login_jsp.**_jspService** (javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServlet...		54.6 ms (1.8%)	25
com.mycompany.hr.process.__EmployeeBeanRemote_Remote_DynamicStub.**addCandidateProfile** (com.myco...		30.8 ms (1%)	20
com.mycompany.hr.process.__EmployeeBeanRemote_Remote_DynamicStub.**authenticate** (com.mycompany.h...		15.2 ms (0.5%)	10
com.mycompany.hr.servlet.HRProcessServlet.**processRequest** (javax.servlet.http.HttpServletRequest, javax...		11.3 ms (0.4%)	22
com.mycompany.hr.process.__EmployeeBeanRemote_Remote_DynamicStub.**addCredentials** (com.mycompany...		8.7 ms (0.3%)	20
org.apache.jsp.AddCandidate_jsp.**_jspService** (javax.servlet.http.HttpServletRequest, javax.servlet.http.Htt...		5.47 ms (0.2%)	23
com.mycompany.hr.servlet.LoginServlet.**processRequest** (javax.servlet.http.HttpServletRequest, javax.servl...		3.21 ms (0.1%)	10
org.apache.jsp.Welcome_jsp.**_jspService** (javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpSer...		1.86 ms (0.1%)	6
com.mycompany.hr.process.EmployeeBeanBean.**authenticate** (com.mycompany.hr.vo.EmployeeCredentials)		1.17 ms (0%)	10
org.apache.jsp.ViewProfile_jsp.**_jspService** (javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpSe...		0.856 ms (0%)	3
com.mycompany.hr.servlet.HRProcessServlet.**doGet** (javax.servlet.http.HttpServletRequest, javax.servlet.htt...		0.368 ms (0%)	22
com.mycompany.hr.dao.BaseDAO.**<init>** ()		0.235 ms (0%)	50
com.mycompany.hr.dao.EmployeeDAO.**<init>** ()		0.195 ms (0%)	50
com.mycompany.hr.process.EmployeeBeanBean.**addCredentials** (com.mycompany.hr.vo.EmployeeCredentials)		0.185 ms (0%)	20
com.mycompany.hr.servlet.LoginServlet.**doPost** (javax.servlet.http.HttpServletRequest, javax.servlet.http.Htt...		0.177 ms (0%)	10
com.mycompany.hr.process.EmployeeBeanBean.**addCandidateProfile** (com.mycompany.hr.vo.CandidateProfile)		0.175 ms (0%)	20
com.mycompany.hr.vo.EmployeeCredentials.**<init>** ()		0.117 ms (0%)	60
com.mycompany.hr.process._EmployeeBeanRemote_Wrapper.**addCandidateProfile** (com.mycompany.hr.vo....		0.114 ms (0%)	20
com.mycompany.hr.process._EmployeeBeanRemote_Wrapper.**addCredentials** (com.mycompany.hr.vo.Employ...		0.109 ms (0%)	20
com.mycompany.hr.vo.CandidateProfile.**<init>** ()		0.104 ms (0%)	40
com.mycompany.hr.process._EmployeeBeanRemote_Wrapper.**authenticate** (com.mycompany.hr.vo.Employee...		0.068 ms (0%)	10
com.mycompany.hr.process._EmployeeBeanRemote_Wrapper.**<init>** (com.mycompany.hr.process._EmployeeB...		0.066 ms (0%)	2
com.mycompany.hr.servlet.LoginServlet.**<init>** ()		0.033 ms (0%)	1
com.mycompany.hr.process.__EmployeeBeanRemote_Remote_DynamicStub.**<init>** ()		0.028 ms (0%)	1
com.mycompany.hr.servlet.HRProcessServlet.**<init>** ()		0.026 ms (0%)	1

Figure 2.Invocations occurred for different components of HR Portal Application

III. Estimation of Reuse Cost Measures

Inorder to estimate the reuse cost measure, it is necessary to know about the number of components available in the related application. The HR Portal application consists of four components as described in Section IV. Initially, the cost of a developing typical system without reuse is considered. It can be represented as follows.

$C_{no\text{-}reuse}$=Cost of developing typical system without reuse

Whenever the reuse is applied to some portion of the system it can be designated as R, the software from a set of component systems.

The Reuse level 'R' can be estimated by considering the number of reused components to the total number of components in the system.

$$\text{Reuse level, R} = \frac{\text{Number of reused components}}{\text{Total Number of components in the system}}$$

$$= 2/4 = 0.5$$

$$= 50\%$$

The Reuse level 'R' usually costs less than developing the whole system from the scratch.

After analyzing the percentage level of reuse components in the system the relative cost to reuse a component has to be defined,

F_{use}= Relative cost to reuse a component

Let us assume that the relative cost to reuse a component is 0.2 as default.

With R=50% and F_{use}=0.2, the cost to develop with reuse is 60% of the cost of developing an application without reuse.

The cost to develop an application system with reuse has two parts. One is the (1-R) part, developed without reuse at the normal cost. The other is the R part, developed with reuse, at a lower cost. In order to do this, it is necessary to estimate the costs separately and add.

$$C_{part\text{-}with\text{-}reuse} = C_{no\text{-}reuse} * (R * F_{use})$$

$$C_{part\text{-}with\text{-}no\text{-}reuse} = C_{no\text{-}reuse} * (1-R)$$

$$C_{with\text{-}reuse} = C_{part\text{-}with\text{-}reuse} + C_{part\text{-}with\text{-}no\text{-}reuse}$$

$$C_{with\text{-}reuse} = C_{no\text{-}reuse} * (R * F_{use} \; (1-R))$$

The cost saved due to reuse

$$C_{saved} = C_{no\text{-}reuse} * C_{with\text{-}reuse}$$

$$= C_{no\text{-}reuse} * (1- (R * F_{use} \; (1-R)))$$

$$= C_{no\text{-}reuse} * R *(1- F_{use})$$

The relative development cost-benefit (ROI) is due to reuse of components is then estimated to be

$$ROI_{saved} = \frac{C_{saved}}{C_{no\text{-}reuse}}$$

$$= R * (1- F_{use})$$

The relative development cost-benefit(ROI) is 40%
When R=50% and F_{use} =0.2 , ROI_{saved} is 40%

It is intended to know how much cost is necessary to know about for creating a new reusable component and manage it. So, this can be denoted as $F_{create.}$

F_{create} = Relative cost to create and manage a reusable component system.

Here all the developed component systems are used to reuse part, R percent, of any application system.

Then the cost to develop the component system for R percent is designated as follows.
$C_{component\text{-}systems} = R * F_{create} * C_{no\text{-}reuse}$

Since F_{create} is much greater than F_{use}, it is necessary to reuse each component and component system several times in several application systems, to make this worthwhile from a cost perspective. It has proved from the literature that the different ranges for F_{create} and F_{use} values depends upon the specific languages, complexity of the problem area and the relevant process followed.

Polin[16] suggested the default values of F_{create} and F_{use} are 1.5 and 0.2.

If there are 'n' application systems in the family, then the cost saving for the application system family is:

$$C_{family\text{-}saved} = n * C_{saved} - C_{component\text{-}systems}$$

$$= C_{no\text{-}reuse} * (n * R*(1- F_{use})- R * F_{create})$$

Finally the return on investment in creating the set of components can be considered as follows.

$$ROI = \frac{C_{family\text{-}saved}}{C_{component\text{-}systems}}$$

$$ROI = \frac{(n * R*(1- F_{use})- R * F_{create})}{R * F_{create}}$$

$$ROI = \frac{(n * (1- F_{use})- R * F_{create})}{F_{create}}$$

When F_{use}= 0.2 and F_{create}= 1.5 then

$$ROI= \frac{(n * 0.8 - 1.5)}{1.5}$$

With breakeven point of minimum value i.e. n > 2**.**

Hence, with the above analysis, the productivity in the organization can be easily improved by increasing the number of the application components which are much reusable.

IV. Conclusion

Component reusability helps in developing quality product as the component in the repository is successfully tested. Most of the reusability metrics proposed in literature or either qualitative or they realize only interface reusability metrics. In this paper an effort was made to propose reusable quantitative measures. The cost for the reused components and non reused components has been quantified. The measures for the family of applications is also estimated. With these cost measures, the budget and effort in the development will get reduced. In future, strategies to measure the generic domain specific components can be quantified.

Acknowledgment

The work was partly supported by the R & D Cell of Muffakham Jah College of Engineering & Technology, Hyderabad, India. The authors would like to thank to all the people from Industry and Academia for their active support.

V. REFERENCES

[1] William B. Frakes, Kyo Kang: Software Reuse and Research: Status and Future, IEEE Transactions on Software Engineering", Vol. 31, No. 7, July 2005

[2] Xichen Wang, Luzi Wang: Software Reuse and Distributed Object Technology, IEEE Transactions on Software Engineering, 2011.

[3] Sametinger: Software Engineering with Reusable Components, Springer-Verlag, ISBN 3- 540-62695-6, 1997.

[4] Jianli He, Rong Chen, Weinan Gu: A New Method for Component Reuse, IEEE Transactions on Software Engineering, 2009.

[5] Maurizio Pighin: A New Methodology for Component Reuse and Maintenance, IEEE Transactions on Softwrae Engineering, 2001.

[6] Yong-liu, Aiguang-Yang: Research and Application of Software Reuse, ACIS International Conference on Software Engineeing, Artificial Intelligence, IEEE, 2007.

[7] N Md Jubair Basha, Salman Abdul Moiz, A.A.Moiz Qyser: Performance Analysis of HR Portal Domain Components Extraction, IJCSIT, Vol. 2(5), 2011, 2326-2331.

[8] Czarnecki, K., Eisenecker, U.W.:Generative Programming: methods, tools and applications. Addison Wesley, London, 2000.

[9] Fuqing Yang, Bing Zhu, Hong Mei : Reuse oriented requirements modeling, Tsinghua University Press, Beizing, 2008.

[10] James F Peters, Witold Pedrycz, " Software Engineering, An Engineering Approach", Wiley India Private Limited, 2007.

[11] Marcus A.S.Boxall, Saeed Araban," Interface Metrics for Reusability Analysis of Components", Proceedings of 2004 IEEE Australian Software Engineering Conference (ASWEC' 04).

[12] Zhongjie Wang et.al, " A Survey of Business Component Methods and Related Technique", World Academy of Science, Engineering and Technology, pp.191-200, 2006.

[13] N Md Jubair Basha, Salman Abdul Moiz, " A Framework Studio for Component Reusability", First International Conference on Information Technology Convergence and Services 2012, pp.325-335,2012.

[14] N Md Jubair Basha, Salman Abdul Moiz, " Model Based Software Development: Issues & Challenges ", IJCSI Vol.II(1,2) 2012, pp.2231-5292,2012.

[15] N Md Jubair Basha, Salman Abdul Moiz, " A Methodolgy to manage victim components using CBO measure ", IJSEA Vol.3(2) 2012, pp.87-96,2012.

[16] Poulin J.S.," Measuring Software Reuse: Principles, Practices and Economic Models" Addison-Wesley.