Software Reusability: Possibilities From The Exisiting Software

Kavita Sharma, Nisha Agnihotri, Minakshi Hooda

Abstract*—Reusability is the likelihood a segment of source code that can be used again to add new functionalities with slight or no modification. Reusable modules and classes reduce implementation time, increase the likelihood that prior testing and use has eliminated bugs and localizes code modifications when a change in implementation is required. Subroutines or functions are the simplest form of reuse. A chunk of code is regularly organized using modules or namespaces into layers. Software reuse is the process of creating software systems from existing software rather than building them from new software. This paper describes the software reuse possibilities and measures how much code can be modified from the existing software? If any problem occurs to the productivity and the comparison of reusable types along with their properties. Finally the reusable software and its cost also discussed.

***Keywords*— Software reuse, Systematic software reuse, software development**

I. Introduction

Software permeates our daily life. There is probably no other human-made material which is more omnipresent than software in our modern society. It has become a crucial part of many aspects of society: home appliances, telecommunications, automobiles, airplanes, shopping, auditing, web teaching, personal entertainment, and so on. In particular, science and technology demand high-quality software for making improvements and breakthroughs. Software Reuse is currently one of the most active and creative research areas in Computer Science. First, we analyze how some design processes, e.g. constructing a problem representation, searching for and evaluating the solution(s), and reuse processes, i.e. retrieving and using previous solution(s), may interact.

Kavita Sharma, Asstt. Professor in Computer Science
DAV Centenary College Faridabad
India
sharma.kavita0033@gmail.com

Nisha Agnihotri, Asstt. Professor in Computer Science
DAV Centenary College Faridabad
India
nis123_agni@yahoo.co.in

Minakshi Hooda , Asstt. Professor in Computer Science
DAV Centenary College Faridabad
India
minakshi_hooda@rediffmail.com

As software systems become more and more complex, software programmers needs to know a variety of information and knowledge in various areas. "Information is wealth", i.e., the knowledge gathered during the development stage can be a valuable asset for a developer as well as the software company. During the software development process, the management and maintenance of knowledge creation is necessary thing. Then only that knowledge is integrated to develop the innovative concept from the older one. So the company must store and manage it for reuse.

A. *History of Software Reuse*

Software reuse began from the origin itself since the programming starts. This is different field in software engineering. However, Doug Mcilroy's paper which proposed basing the software factory on reusable components. In late sixties software crisis developed and managed these types of researches. In 1970s Academia proposed some models of research regarding reuse management in software field. After that in 1980s large scale reuse programs are done, in this stage it reach the all programmers to known that reusability saves time and cost. The following shows that various peoples tell the definition for Reuse:

"Reuse is the use of any information which a developer may need in the software creation process" (Freeman, 1987)

"The use of everything associated with a software project, including knowledge" (Basiliand Rombach, 1988)

B. *Software Reuse*

What is Software Reuse? Its process of creating software system from existing software assets rather than building software system from scratch. The development of new software from the existing one. The modification or alteration of the existing software into the new software. The known concept which was used to integrate the innovative concept. The Assets can be software components, software requirement analysis manuals, and design models, database schema, objects, code documentation, domain architecture, standards, test scenarios, and plans. The existing software can be from within a software system or other similar software systems or widely in different systems. For example, Ms Office 2003 a tool to create and to edit different types of documents, worksheets, slides and databases. They came up

with the Ms Office 2007 which is the latest version of it. Just like this there are so many examples we can consider as a Software Reuse. It is hard to develop the software that achieves Reliability, Portability, Extensibility, Flexibility, predictability and Efficiency and very hard to developing systematically high quality reusable software components and framework. Those Reusable workings and frameworks are naturally summary, which makes it hard to engineer their we use Software Reuse? The Proper Reuse of software process leads to increase the quality of product, increases the reliability, productivity improvement, and reduces time to market, reduce the cost of developing the product. It's not a myth. In short, the development of a reuse process and repository produces a base of knowledge that minimizing the amount of development work required for future projects, improves in quality after every reuse and ultimately reducing the risk of new projects that are based on repository knowledge.

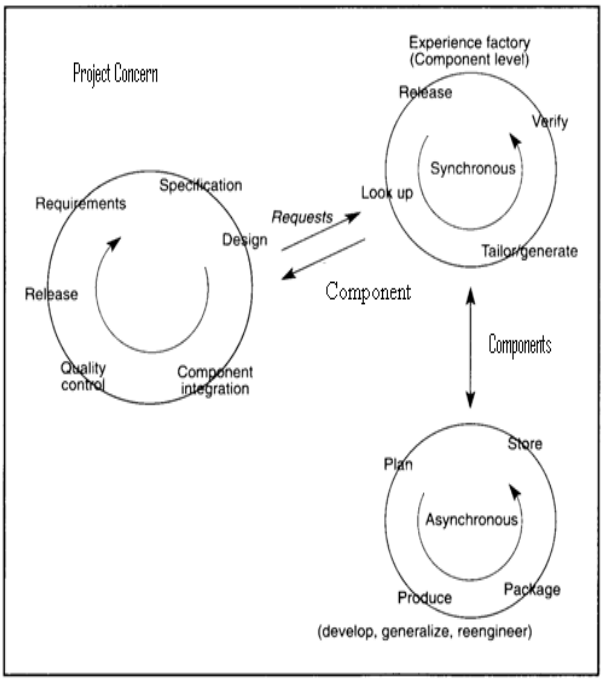

Figure 1. Reuse Process Model

The above figure shows an organizational framework the divides project's reuse packaging activities from the project-specific activities, which process models support the each activity. The framework defines two separate organizations: a project concern and an experience factory. The project concern develops the product which was get source from the packaged experience from prior and current developments. In turns, the project offers its own experiences to other projects. i.e., this developed project can act as existing source for another new project. The experience factory stores the current aspects as well as the existing packaging for use it again.

- 40% to 60% of code is reusable
- 60% of design and code are reusable in
- business application
- 75% of program functions are common
- 15% of the code is unique
- 15% to 85% -rates of actual and
- Potential reuse

C. *Generation Vs. Composition*

TABLE1. GENERATION VS COMPOSITION

Reuse Techniques	**Composition**	**Generation**
Reused component	Building Blocks	Patterns
Nature of component passive	Atomic and immutable	Diffuse and malleable, active
Emphasis	Composition Principles Generators	Language-based Generators
Examples	Function/class Libraries, Unix filters	4th generation Languages Parser generators

The above Table inform us that the difference between the composition and generation under the reusing techniques such as Reuse component, Nature of the component, Emphasis, and their corresponding examples.

D. *Framework for reuse and flexibility (Object oriented)*

OOPs are the new concept of programming parallel to Procedure oriented programming. It was introduced in late 80's.It considers the programming simulated to real world objects. It helps in programming approach in order to built robust user and limit friendly and efficient software and provides the efficient way to maintain real world software. Overall, three essential concepts comprise object technology: information hiding, inheritance/polymorphism and dynamic binding. These concepts are combined and create new techniques of object technology. Moreover the object-based and object-oriented programs are not automatically modified. The programmer should modify them for create the new concept. The coupling of components1 often manifests in the source code. Figure 2 schematically outlines the problem. If the right hand component should work with another component, its source code has to be changed.

E. *Reusable Software*

- Commercial off-the–shelf-COTS: The software produced under this category meets the user's requirements exactly. It is optional of using additional functions which was added to that the existing software. The various parts of the COTS components were supplied by its suppliers. The additional attachments are tested and documented for the user reference. They are stand alone applications such as Visual basic TM and Sybase TM.
- GOTS (Government off-the-shelf): This is government development software for their own purpose and it is not for sale. This is very similar to the COTS application except it is with / without documentation (not compulsory).Moreover it is occasionally updated and improved. Mostly it deals with GFI (Government Furnished Information) only.
- Planned Reuse: The Planned reuse software is somewhat differed from other application, it is planned and designed from the existing software. Moreover it finds the existing software's drawbacks and provides solution in the current developing software. So it is always created with documentation for assist the users. Sometimes it is updated and improved from the current state.
- Incidental Reuse: The incidental reuse is not a standard option for creation of reuse software. This is only planned at the time of cost balance situation. Moreover it is not improved or updated. The design for reuse is rarely in this concept, mostly these type of software developed in very low duration.

TABLE 2. COMPARISON OF TYPES OF REUSABLE SOFTWARE

	Comme--rcial off-the–shelf	**Government off-the-shelf**	**Planned Reuse**	**Incidental Reuse**
Documentation and Ready Status	yes	sometimes	Often	sometimes
Balancing the Development cost	often	often	often	often
Modification & Open Standard reference	often	often	Sometimes	occasionally
Reuse Design & testing	usually	often	sometimes	occasionally
Updating and Improvement	usually	occasionally	sometimes	seldom

The above table outlines the characteristic differences among types of reusable software. The various reusable software are categorized by its using areas and time which it was developed. They are categorized as COTS, GOTS, Planned Reuse and Incidental Reuse. They are differentiated under the topics such as Documentation and Ready Status, Balancing the Development cost Modification & Open Standard reference Reuse Design & testing Updating and Improvement.

F. *Conclusion and Future work*

Define abbreviations we conclude that reusing of software is emerging technology, which saves the production cost, improving the innovative technology from the older one. So if we go for wrong track in new thing means, we return back with the older concept. Reuse research has been ongoing since the late 1960s and domain engineering research since the 1980s. Much has been accomplished, but there is still much to do before the vision of better system building via reuse and domain engineering is completely achieved. Though most organizations reuse components to save the time and cost, reuse is never risk free.

References

[1] Arun Sharma, Rajesh Kumar, P S Grover "Managing Component-Based Systems With Reusable Components" International Journal of Computer Science and Security, Volume 1 : Issue (2)

[2] Bruno Antunes, Paulo Gomes and Nuno Seco "SRS: A Software Reuse System based on the Semantic Web" Universidade de Coimbra 3030 Coimbra.

[3] Capers Jones "Best Practices for Certifying Reusable Material " Version 4.0 December 18, 2008

[4] Dan Galorath "Software Reuse and Commercial Off-the-Shelf Software
Magazine", 1995. 5. Bradford, Kathy and Lori Vaughan. Improve Commercial-off- the-Shelf (COTS) Integration Estimates. Redondo Beach: Northrup Grumman Mission Systems, 2004.

[5] Eng Huat Ng "Software Reusability and its Application to Interactive Multimedia Learning System", Byrom Street, Liverpool L3 3AF, UK 1998

[6] Gary Boetticher David Eichmann "A Neural Network Paradigm for Characterizing Reusable Software ",NCC-9- 16, RICIS research activity number RB12 ACOSM'93, Nov.18-1993

[7] Gianluigi Caldiera and Victor R.Basili University of Mary land "Identifying and Qualifying reusable software components ", 0018-916/91/0200-0061 IEEE Feb 1991

[8] Parvinder Singh Sandhu, Janpreet Singh and Hardeep Singh Approaches for "Categorization of Reusable Software Components", Journal of Computer Science 3 (5): 266-273, 2007 ISSN 1549-3636

About Author (s):

Kavita Sharma
Asstt. Professor in Computer Science,
DAV Centenary College, Faridabad,
India