A Study on Risk Assesment in Software Design Reusability

Chethana. S
Computer Science Department
Nmkrv Pu College For Women
Jayanagar 3rd Block
Bangalore-560011

Dr. G. N. Srinivasan
Professor3
Dept.Of Information Science
And Engineering R.V.Vidyaneketan
Post,Mysore Road
Bangalore-560059

***Abstract* : Software reusability is commonly explained in the software development disaster. When we try to apply the solution to solve a problem, it becomes necessary to make the work easy and simple. Software reuse has become important in case of software design due to its possible benefit which includes increased product quality and decrease product cost and schedule. Software reuse is none other than the active software or software components to construct a new software. The idea of reuse is nothing but the skill to merge the software concepts to form a larger unit of software. It improves the quality and productivity of software production process. This paper in brief gives the review of the current research status in the field of software reuse and the research contributions. Here several upcoming trends for research in software reuse are examined[1].**

INTRODUCTION :

Software reuse is a basic aspect of high quality software. Efficient reuses of software products increases the productivity by saving time and reduce the cost of software development. The concept of reusing software components is clear at the code level. The same concept is difficult to analyze in the context of reusing designs. Basically science and technology demands for a high quality software for further enhancement.

TYPES OF REUSE:

a. Opportunistic reuse: Despite the fact that realizing to begin a project, it is necessary to understand the different existing components that can be reused.
b. Planned reuse: Here it is needed to design the components so that they can be made reusable in future development.

Opportunistic reuse can be classified as :

a. Internal reuse: It reuses its own components. This might be a business decision. In this case the team takes initiative to have a control over the component and becomes critical to the project[2].

b. External reuse: A third party component which choose a license. A third party licensing costs 1 to 20% of what it would cost to develop internally. It is necessary to consider the time to analyse and combine the component.

Category of reuse in software engineering.

Application system reuse : the entire application system can be reused. This can be done by interpreting it without change into other system[3] (cots reuse) or by expanding the application families.

Component reuse : components from sub systems to single objects may be reused.

Object and functional reuse : software modules which executes a well defined object or function can be reused.

SOFTWARE REUSE ADVANTAGES.

Increased dependability : Reuse software which has been used and tested with in the systems working are more dependable than the new software. The basic use of the software leads to any design and allow implementation faults. They are fixed, by further reducing the number of failures when the software is reused.

Reduces process risk :In case if software is present, there would be a dinama in the costs of reusing the software rather than in the cost of development. This is a major factor in case of project management where it reduces the error in project cost estimation. This is true only when a large software components such as subsystems are reused[4].

Effective use of specialists: As a substitute of an application specialists have done the same work on different projects (3,4). These specialists can extend the reusable software which summarizes their knowledge[5].

Standards compliance: Some models, such as interface models, can be applied as a set of standard reusable components. For example- if menus in a user interface are employed by reusable components, hence all application present the same menu formats to the users. It improves dependability because users less likely make mistakes when presented with a known interface.

Accelerated development: Fetching a system to market as fast as possible is often more important than the over all development cost. Reusing software can speed up the

system production as both development and validation time should be decreased.

Reuse challenges: In case the source code of a reused software system or component is not present then upholding cost may be increased as the reused elements of the system may become increasingly incompatible with system changes.

Lack of tool support: The cast tools sets may not sustain with reuse. It may be complicated to integrate these tools with a component library system. The software process may be believed by these tools which may not take reuse into account.

Not invented here -syndrome: some software engineers occasionally choose to re write components as they consider that they can improve on the reusable components. Here writing the original software is more likely tested that reusing other people's software.

Creating and maintaining a component library: sustaining a reusable component library and make sure that the software developers can use this library may be expensive. Our present methods for classifying, cataloguing and retrieving the software components are undeveloped.

Finding, understanding and adapting reusable components: A Software component has to be realized in a new background. Engineers must be sure of finding a component in the library and make use of a component normally as a part of their development process.

The reuse landscape: Though reuse is considered as the reuse of system components, there are dissimilar types of reuse that may be used. Reuse is feasible at variety of levels from simple functions to complete application systems. It covers the range of portable reuse techniques[6].

Reuse approaches: The specific ideas which arise across applications are signified as design patterns which show conceptual and existing objects and interactions.

Component based development: The systems are expanded by combing the components. Which match to form a component model standard.

Application framework: A group of conceptual and existing classes are developed to create application systems.

Legacy system wrapping: Legacy systems are covered by defining an interface and by providing access to these legacy systems through the interfaces[7].

Service –oriented systems: Systems are built up connecting shared services that may be externally provided.

Application product line: An application type is a wide spread common architecture which may be adapted in different ways for different customers[8].

COTS integration: Systems are extended by combining the existing application systems.

Configurable vertical applications: A basic system is proposed to configure the needs of specific system customers.

Program libraries: Class and function libraries apply familiarly used abstractions which are available for reuse.

Program generators: A system generator creates awareness of a scrupulous type of application which can generate systems or system fragments in that domain[8].

Aspect-oriented software development: Shared components are used in to an application at different places when the program is complied[9,10].

CONCLUSION:

Software reuse is considered as a key to progress the software development productivity and quality. There is lot of research going on in case of software reuse as a part of software development life cycle. This paper brings out the important aspects of software reuse research. In this paper it has been brought out the different types of reuse approaches and challenges. Further research on software reuse has to be addressed. Where it meets the needs of industry and also the customers' needs.

ACKNOWLEDGEMENTS:

I am grateful to my guide Dr. Srinivasan ,Sri Rashmi for their support in bringing out this paper successfully. I would especially thank Dr. Suchithra madam also for giving her time and input.

REFERENCES:

[1] Tawfig.M.Abdelaz.Z, Yasmeen.N.Zada and Mohamed.A.Hagal (2014) A structural approach to improve software design reusability.
[2] Software engineering, VOL SE-12 ,no. 1 1994. Gert B (1988) Morality, Oxford University press.
[3] Green R.M (1994) The Ethical Manager, Macmillia publishing.
[4] Gotterbam and Rogerson 1998, "The Ethics of Software Project Management" in Ethics and Information Technology ".ed Gsan Collste, New academic publisher,1998.
[5] Humphrey,W.A. Discipline of Software Engineering Addison Wesely Longman, Reading Mass,1995.
[6] Linger R.A Clean room process Model @IEEE Software March 1994.PP 50-58.
[7] Smith 199 Co.U.Smith performance Engineering of Software Systems, Reading ,M.A.Addison –Wesley 1990.
[8] Smith and Williams 2002 C.U.Smith, performance solutions : A practical guide to creating responsive , scalable software , Boston M.A, Addison Wesley 2002.
[9] Williams and Smith 2002 a.L.G.Williams and C.U.Smith ,"PASASM" A method for the performance Assessment of Software Architectures" 2002.
[10] Sarbjeet Singh, Sukhvinder Singh,Gurpreet Singh."Reusability of the Software" vol 7-no, 14 2010.