International Journal of Computer Science and Engineering Research and Development (IJCSERD), ISSN 2248-9363 (Print), ISSN 2248-9371 (Online),
Volume 1, Number 1, April-June (2011)
pp. 10-14 © PRJ Publication
http://www.prjpublication.com/IJCSERD.asp

IJCSERD

© PRJ PUBLICATION

CHALLENGES FACED DURING SOFTWARE COMPONENT SELECTION

Arvind Kumar, Pradeep Tomar and Deepak Panwar
School of Information and Communication Technology
Gautam Buddha University, Greater Noida-201 308, Uttar Pradesh, INDIA
k.arvind33@gmail.com; parry.tomar@gmail.com

ABSTRACT

Software development organizations believe that Component-Based Software Engineering (CBSE) approach will improve their productivity and quality by selecting pre-existing software components. CBSE is an approach which is used to enhance the reusability because reusability is a way to improve efficiency and productivity of software systems. Component-Based Software Development (CBSD) with reusable component not only reduces the time to market but also brings down the cost of development heavily. This paper presents the challenge like performance, time, components size, fault tolerance, reliability, components functionality, components compatibility and available component subset which are faced by developer during component selection. Lastly this paper summarizes which algorithm is used for component retrieval according to availability of component subset by keeping this challenges in mind.

1. INTRODUCTION

Software component selection is considered as an important solution to many of the problems in Component-Based Software Development (CBSD). In the early 1990's, it became apparent to both researchers and practitioners that object-oriented technologies were not enough to cope with the rapidly changing requirements of real-world software systems. So researchers and practitioners decide to shift towards the component technology. If we had a collection of reusable software components, we could build applications by simply plugging existing components together. In CBSD, a complex system is accomplished by assembling simpler pieces obtained in various manners. The research efforts have been made to make reusability process of component-based software more effective, more predictable and less expensive in comparison to simple software reusability by using different approaches. CBSD and Component-Based Software Reusability (CBSR) approaches of software engineering is not similar to traditional engineering domain. CBSD and CBSR finally provide solution to all complex problems and not only reduce the time to market but also bring down the development cost heavily [1]. CBSE is an approach which is used to enhance the reusability from the pre-existing software components. But when reuse pre-existing software components, components selection factors play an important role to enhance the reusability, productivity and quality. This paper presents analysis of challenges faced during components selection to suggest how to select component due to which researches and practitioners select optimal components from repository to fulfill the requirements of client.

2. CHALLENGES FACED DURING SELECTION OF COMPONENT

In components selection, a number of software components selected from a subset of components or from components repository in such a way that their composition satisfies a set of objectives. So this paper analyzes challenges of components selection which help to select optimal components from component repository in CBSD. Components selection factors play an important role in CBSE. This paper presents the following challenges like performance, time, components size, fault tolerance, reliability, components functionality, components compatibility and available component subset for consideration during the software component selection. If researcher and practitioners keep all the challenges in mind during component selection, client can be able to get good quality software.

2.1 Performance

Performance of software component very helpful for in maintaining the quality of software during software development, so performance is a main challenge during the selection of component. The degree to which a system or component accomplish its designed function within given constrains such as accuracy, availability, efficiency, response time, recovery time, resource usage, speed etc. [2]. Performance can be increased by selecting those components which contains high cohesion, less coupling and less number of interfaces of components according to the following equation.

Performance $\propto$ 1/Interface $\propto$ Cohesion $\propto$ 1/ Coupling

Coupling is a major challenge during component selection because coupling between components is the strength of interconnection between components or a measure of interdependence among components [6]. Coupling is the interdependence between components into other components. If researchers and practitioners analyze the coupling complexity of components and choose the component with lower coupling so these components are beneficial for integration.

Coupling $\propto$ 1/Performance $\propto$ Interface $\propto$ 1/ Cohesion

Cohesion is also a major challenge during component selection because cohesion of a component represents how tightly bound the internal elements of the components to another [6]. So if researchers analyze the cohesion complexity and choose the components which have higher cohesion so these components are beneficial for integration.

Cohesion $\propto$ Performance $\propto$ 1/ Coupling $\propto$ 1/Interface

2.2 Time

Software development activities require more time for best software quality. But client want to reduce the development time. So this study proposes to use maximum number of COTS component because COTS components save the development and testing time and improve quality according to the following equation in which time and COTS are inversely propositional to each other.

Time $\propto$ 1/COTS

Commercial off-the-shelf component describes software or hardware products that are ready-made and available for sale to the general public. The use of COTS is increasingly becoming

commonplace. This is mainly due to reducing budgets, accelerating rates of COTS enhancement, reducing development time and effort constraints, and expanding system requirements [5].

COTS $\propto$ 1/Time $\propto$ 1/Cost

2.3 Components Size

Components size depends on programming language and components code may be written in high level or low level language. User wants the size of system should be less. So researchers and practitioners select those components which use high level language. It means select those components which depend on high level language according to the following equation in which size and high level language are inversely propositional to each other.

Size $\propto$ 1/ Programming Language (High Level Language)

2.4 Fault Tolerance

The ability of system or component to work for long time continuously without any hardware or software faults [2]. Fault tolerance can be increased by increase in Mean Time to Failure (MTTF) according to the following equation in which fault tolerance and mean time to failure directly propositional to each other. Fault-tolerance or graceful degradation is the property that enables a system to continue operating properly in the event of the failure of some of its components

Fault Tolerance $\propto$ Mean Time to Failure

2.5 Reliability

The ability of a system or component to perform its required functions under stated conditions for a specified period of time [2]. Reliability can be measure in terms of reliability metrics MTTF, MTTR, MTBF, ROCOF and availability. This paper gives here main focus on software availability. Software availability means a program should be accessible according to requirement at a given point of time. So reliability improves the performance and availability of the components according to the equation in which reliability and availability are directly propositional to each other.

Reliability $\propto$ Availability

2.6 Reusability

The degree to which a software component can be used in more than one computer program or software system [2] is a main challenge to select a good quality software component. CBSE is an approach which is used to enhance the reusability with the development of CBS from the pre-existing software components and reusability save the development time, effort and cost. If researchers and practitioners use only reusable components it will be beneficial for software industry.

2.7 Components Functionality and Architecture

Component functionality and component architecture are the most serious challenges of an existing component that cannot be reused without change. Existing component and a component to be newly developed match according to their functions and architecture. The new component to be developed may require some changes to corresponding functions and component architecture in the existing component or may require additional functions. Suppose the existing component contain excessive function, but to increase the performance most of the excess

function may need to be eliminated [3], [4]. So try to select component according to customer requirement.

2.8 Component Compatibility

In CBSD the most important challenges for successful reusability of component is the compatibility between different versions of the components. A component can be replaced easily or added in new parts of a system if it is compatible with its previous version. The compatibility requirements are essential for running software system, for many years. Compatibility issues are relative simple when changes introduced in the software systems are of maintenance and improvement nature only. Using appropriate test plans, including regression tests, functional compatibility can be tested to a reasonable extent [3], [4].

2.9 Available Component Subset

Available component subset inside the repository is also a main challenge during component selection according to the component requirements for a specific application domain. Available component subset may be small, moderate and large which create a problem for selection of algorithm for component retrieval.

3. ANALYSIS OF COMPONENT SELECTION ACCORDING TO AVAILABILITY OF COMPONENT SUBSET

There are so many challenges but this paper is going to present main focus only on available component subset, it may be small, moderate, and large in software component repository. So according to there availability researchers and practitioners can use the following algorithms for component selection.

3.1 Available Component Subset: Small

If available component subset is small in components repository then complete enumeration method can use for optimal selection [7]. Complete enumeration method has all possible combinational solution. Researcher see the best objective function value among all available component subset then select the best component subset.

3.2 Available Component Subset: Moderate

If available components subset is moderate in components repository then researcher can use Longest Common Subsequences (LCS) and Greedy algorithm to select for optimal components subset from repository. LCS determines the LCS of two sequences (subset) of items [8]. It means, there are two components subset, one components subset is a requirement components subset and another components subset is available components subset in repository. It may be possible there are many components subset which are available for reuse in components repository. So researchers and practitioners compare requirement subset and each available components subset and determine the LCS. Hence which components subset will give high LCS that components subset select from components repository for target system. Greedy algorithm is playing main role in component selection. Greedy algorithm looks for locally optimal solution and assumes it as best but they do not always yield the optimal solution but it never backtracks or changes past choices. Greedy techniques are used to find optimum components and use some heuristic to generate a sequence of sub-optimums that hopefully converge to the optimum value. Once a sub-optimum is picked, it is never changed nor is it re-examined [9], [10], [11].

3.3 Available Component Subset: Large

If available component subset is large in component repository then single pass heuristic and genetic algorithm can be use [7]. Single pass heuristic is developing an efficient and unique to the

problem situation and will give the solution after performing a polynomial number of steps. Hence, the solution through such heuristic will be local optimum solution. Genetic algorithm is a meta heuristic aims at a global optimum which will be almost closer to the optimum solution [11].

4. CONCLUSION

Researchers and practitioners keep in mind the above challenges and algorithm during component selection because they not only improve the optimal component selection and productivity but also put a positive impact on the quality and maintainability of software products. Analysis of challenges before component selection support development of good quality software. Initially, the software engineers analyze these challenges to select optimal components from pre-existing reusable component repository. In addition, a repository should address the problem of available component subset that are of different in term quantity, therefore according to the analysis this paper discusses, which algorithm is best for optimal component retrieval according to the availability of component subset. These major challenges are very helpful for researcher, software developer and the large software development organization for productivity enhancement of CBSD.

5. REFERENCES

[1] Gill, N. S. and Tomar, P. (2007). "Software Component Technology: An Easy Way to Enhance Software Reusability", proceedings of *94th Indian Science Congress Conference*, Annamalai University, Tamilnadu, INDIA, *pp. 18-19.*

[2] IEEE Standards Board (1990). "IEEE Standard Glossary of Software Engineering Terminology", *Computer Society of the IEEE.*

[3] Gill, N. S. and Tomar P. (2006). "Major Challenges Faced in Component-Based Software Reuse", *Journal of Computer Science,* Vol. 2, No. 01, *pp. 53-58.*

[4] Gill, N. S. and Tomar, P. (2007). "CBS Testing Requirements and Test Case Process Documentation Revisited", *ACM SIGSOFT Software Engineering,* Vol. 32, No. 02, *pp. 1-4.*

[5] www.defenceiq.com/glossary/cots-components.

[6] Jalote, P. (2005). "An Integrated Approach to Software Engineering", 3rd Edition, *Narosa Publishing*

[7] Panneerselvam, R. (2010). "Research Methodology", *PHL learning private limited.*

[8] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L. and Stein, Clifford, (2001). "Introduction to Algorithms", 2nd Ed., *MIT Press and McGraw-Hill*

[9] Haghpanah N.; Moaven S.; Habibi J.; Kargar M. and Yeganeh S. H., (2007). "Approximation Algorithms for Software Component Selection Problem", proceedings of *APSEC, IEEE Computer Society*, *pp. 159–166.*

[10] Fox, M. R.; Brogan, D. C.; Paul, J. and Reynolds, F. (2004). "Approximating Component Selection", proceedings of the *36th conference on Winter Simulation Conference*, *pp. 429-434.*

[11] Andreea Vescan, (2008). "Pareto Dominance - Based Approach for the Component Selection Problem", *Second UKSIM European Symposium on Compute*, *pp. 58-63*