CHAPTER 4:- CHALLENGES OF COMPONENT SELECTION
Software component selection is considered as an important solution to many of the problems in CBSD. In the early 1990’s, it became apparent to both researchers and practitioners that OOT had not fulfilled all functionality software systems. So researchers and practitioners decide to shift towards the component technology. If we had a integration of reusable software components, we could build applications by simply plugging existing components together. In

The research efforts have been made to make reusability process of CBS 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 [Pressman S. R. (2006)].

4.1 CHALLENGES OF COMPONENT SELECTION

This chapter 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 chapter 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.

4.1.1 Performance

Performance must be enhanced by selection those software modules and components which contains less coupling, high cohesion, and less number of interfaces.

\[
\text{Performance} \propto \frac{1}{\text{Interface}} \propto \text{Cohesion} \propto \frac{1}{\text{Coupling}}
\]
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 [Jalote, P. (2005)]. **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.

$$\text{Coupling} \propto 1/\text{Performance} \propto \text{Interface} \propto 1/\text{Cohesion}$$

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 [Jalote, P. (2005)].

$$\text{Cohesion} \propto \text{Performance} \propto 1/\text{Coupling} \propto 1/\text{Interface}$$

So if researchers analyze the cohesion complexity and choose the components which have higher cohesion so these components are beneficial for integration.

**4.1.2 Time**

Programmer wants to minimize the development time. So researcher used the reusable components, COTS component.

$$\text{Time} \propto 1/\text{COTS}$$

**4.1.3 Component Size**

Every programmer wants to reduce the components size, but it depends on programming language.

$$\text{Size} \propto 1/\text{Programming Language (High Level Language)}$$
4.1.4 Fault Tolerance
The ability of system or component to work for long time continuously without any hardware or software faults [IEEE].

| Fault Tolerance \(\propto\) Mean Time to Failure |

4.1.5 Reliability
The ability of a system or component to perform its required functions under stated conditions for a specified period of time [IEEE].

| Reliability \(\propto\) Availability |

4.1.6 Reusability
If researchers and practitioners use only COTS or reusable software components it will be beneficial for software industry.

4.1.7 Components Functionality and Architecture
Suppose the existing component contain excessive function, but to increase the performance most of the excess function may need to be eliminated [Gill, N. S. and Tomar P. (2006)]. So try to select component according to customer requirement.

4.2 COMPONENT SELECTION ACCORDING TO COMPONENT SUBSET
There are so many challenges but this chapter is going give main focus only on available component subset, it may be small, moderate, and large in software component repository. So according to their availability researchers and practitioners can use the following algorithms for component selection.

4.2.1 Available Component Subset- Small
If available component subset is small in components repository then complete enumeration method can use for optimal selection [Panneerselvam, R. (2010)]. Complete enumeration method has all possible combinational solution (possible component subset) and there can be solve optimally each one. Researcher see the best objective function value among all available component subset then select the best component subset.
4.2.2 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 [Cormen.(2001)]. 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.

4.2.3 Available Component Subset- Large

If available component subset is large in component repository then single pass heuristic and genetic algorithm can be use [Panneerselvam, R. (2010)]. 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. The genetic operators are used crossover and mutation [Vescan A. (2008)].

Researchers and practitioners keep in mind the above challenges during component selection because they only improve the optimal component selection and productivity. 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 chapter 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.