International Journal of Science and Engineering Applications
Volume 4 Issue 2, 2015, ISSN-2319-7560 (Online)

Issues and Concerns In Software Component Selection

Nitish Madaan
Department of CSE and IT
ITM University
Gurgaon, India

Jagdeep Kaur
Department of CSE and IT
ITM University
Gurgaon, India

Abstract: The increasing availability of COTS (commercial-off-the-shelf) components in the market of software development has concretized the opportunity of building whole systems based on previously built components. Component-Based Software Engineering (CBSE) is an approach which is used to improve efficiency and productivity of software system with the help of reusability. CBSE approach improves software development productivity and software quality by selecting pre-existing software components. Reusability in Component-Based Software Development (CBSD) not only reduces the time to market in development but also brings the cost down of development heavily. This paper represents the challenges which are faced by software developer during component selection like reliability, time, components size, fault tolerance, performance, components functionality and components compatibility. This paper also summarizes algorithms used for component retrieval according to availability of component subset.

Keywords: Component Based Software Development (CBSD), Software Component Selection, Case Based Reasoning, Component Based Software Engineering (CBSE), Software selection process.

1. INTRODUCTION

As compared with traditional approach, Component-based software engineering (CBSE) helps in developing a quality software system with less time and less resources. It is necessary to identify the software component and must be evaluated in order to check if they able to provide the requisite functionality or not for the system under the development. Most of the problems in the component based software development are considered to be solved as the efficient component selection.

In the early 1990's,researchers and practitioners choose to shift towards the component technology because it became visible to both researchers and practitioners that object-oriented technologies were not enough to manage with the rapidly changing requirements of real-world software systems. If we had a collection of reusable software components, we could build applications by simply plugging existing components together In Component Based Software Component, a complex system is build with the help of assembling the simpler and small pieces obtained in various manner. By using different approaches research efforts have been made to make the process of reusability of component based software more effective, predictable and less expensive as compare with simple software reusability. CBSD and Component-Based Software Reusability (CBSR) approaches of software engineering is not similar to traditional engineering domain. CBSD and CBSR at last provide solution to all complex problems and help not only to reduce the time to market but also help in bring down the development cost significantly [1]. During reuse of pre-existing software components, components selection factors play an important role. CBSE is an approach which is used to enhance the reusability from the pre-existing software components. Already built software component selection process identifies methods to extract software requirements in the general sense, but they do not explicitly address how to specify security requirements [4].

Another challenge is Security requirements specification because requirements cover both functional and non-functional aspects and many software developers may not be familiar with the scope of security issues needing to be addressed. This paper presents analysis of challenges faced throughout the software component selection.

2. SOFTWARE COMPONENT SELECTION ISSUES

In components selection, a number of software components selected from a subset of components or from components repository in a manner so that their composition satisfies the specifications. In CBSE component selection factors plays an important role. Client may get very good quality software, if researcher and Practitioners keeps all the challenges in mind at the time of selection of components.

2.1. Software Quality Evaluation

There are a number of quality attributes are concerned for the development of a software system. We may define the overall system quality measure (Q), based on a set of quality attributes (A) as proposed by Vows and Arrest [2]. The set attributes (A) includes reliability, performance, fault tolerance, safety, security, availability, testability, and maintainability. In a straightforward manner some other measurable quality attributes considered as important may also be included.

Overall software quality may be understood as weighted linear combination of the values for each of these attributes:

$$Q= w_R R + w_P P+ w_F F+ w_{Sa} Sa+ w_{Se} Se+ w_{Av} Av+w_T T+ w_M M \quad (1)$$

Where,
R = Reliability
P = Performance
F = Fault tolerance
Sa = Safety
Se = Security
Av = Availability
T = Testability
M = Maintainability

The values w_{R}, w_{P}, w_{F}, w_{Sa}, w_{Se}, w_{Av}, w_{T}, w_{M} denote the weights assigned to the corresponding quality attribute. The summation of all the weights assigned to all the quality attributes is equal to 1. This approach facilitates a simple, flexible, and consistent way to evaluate and compare the total software quality of proposed designs based on the needs of the stakeholders. The type of software describes the weighting for each attribute. The weight for security would probably be higher for a financial system than that for safety while the weight may be less for testability. The weighted key attributes for a safety critical system, would maybe be reliability, performance, safety, fault tolerance, and availability. The key weighted attributes for an ecommerce system would be reliability, performance, availability, security, and maintainability. If we understand each candidate component to contribute a certain amount of value toward each individual quality attribute then this approach may be extended for use within a component-oriented context. However, each quality attribute has its own unit of measurement.
Comparison of different types of measurement of units is not able to be compared in a significant manner. In order for the weighting scheme described above to make meaningful comparisons it is necessary for all quality metrics must be able to be of equal scale. For example, Maintainability complete of 4.5 and a Reliability score of 4.5 should both contribute equally toward the overall software quality if their weights are equal. To measure all quality attributes for the component to compare appropriately, it is necessary to standardize all quality attributes.

2.2. Components selection factors

Selection of a components are dependent on some factors that are identified as performance, time, size of component, fault tolerance, functionality of components , reliability, compatibility of components and also considered the availability of component subset.

a) Performance

At the time of selection of component performance is the main challenge. Performance of a system cannot be expressed in terms of the performance of its individual component. Performance is the extent 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. Performance cannot be calculated for the individual component, rather than it is calculated for the completed system after integration of the system. To increase the performance of the system select those software components containing high modules cohesion , less module coupling, and less number of interfaces of components.

b) Time

Development time and testing time saves when we use COTS components and it also improves the quality of our software.

c) Size of Component

Size of the components completely depends upon the programming language and the code of components that may be written in low level or high level languages. User of the system always wants that the size of the system should be less. So to achieve this it is important point to be noticed that on high level language written components must be used because the requires lesser size.

d) Fault Tolerance

Increase in Mean Time to Failure (MTTF) helps in increasing the fault tolerance. Capability of a component or module to run continuously without having any fault in any type of software or hardware[3]. 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.

e) Reliability

The capability of component or system to execute its required functions under stated conditions for a specified period of time [3]. Reliability metrics helps in the measurement of the reliability like MTTF, MTTR, MTBF, ROCOF and availability. Reliability and availability are directly proportion to each other, where performance and availability of the components are improved by reliability.

f) Reusability

The degree to which a software component can be used in more than one computer program or software system [3] 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 preexisting software components and reusability save the development time, effort and cost.

g) Components Functionality and Architecture

The most serious challenges of an already built component is that functionality of the component and the architecture of component cannot be reused anywhere without change. According to the functionality and architecture of the component needs to be matched of newly develop component with the existing components that are already built. Some of the changes may require in the existing component functions and component architecture along with some extra features to develop the new component.

h) Compatibility of Components

Checking compatibility between various versions of the components is most important challenge in the successful reusability of component. Already built component can be easily replaced or can be added in a new part easily if it is compatible with the previous version. For many years, the compatibility requirements are essential for running software system. Compatibility issues are relative simple when changes introduced in the software systems are of maintenance and improvement nature only. In a reasonable extent it is necessary to test plans, including regression tests, functional compatibility.

i) 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.

2.3. Common Steps of Component Selection Methods

Although there is no commonly accepted method for component selection, all methods share some key steps that can be iterative and overlapping [4]. These steps are described as follows:

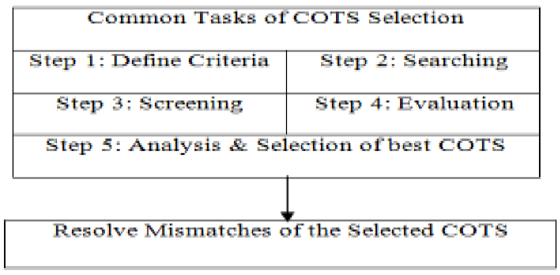
Common Tasks of COTS Selection
Step 1: Define Criteria
Step 2: Searching
Step 3: Screening
Step 4: Evaluation
Step 5: Analysis & Selection of best COTS
Resolve Mismatches of the Selected COTS

Figure 1. The general COTS selection process [5]

Step1: Define the evaluation criteria

Step2: Search for components.

Step3: Filter the search results based on a set of must have requirements. This results in defining a short list of most promising component candidates which are to be evaluated in more detail.

Step4: Evaluate components.

Step5: Analyze the evaluation data and select the components that have the best fitness with the criteria.

3. CONCLUSION

This work emphasizes on how to address the issue of component selection based on component costs and quality dimensions. Selection of optimal set of components not only improves selection process of component but also has a positive impact on the searching a good quality software component for software development. Addressing Components selection factors in an effective way will not only improve the optimal component selection and productivity but also put a positive impact on the quality and maintainability of software products. 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.

4. FUTURE SCOPE

The field of quality attribute determination of component-based system is extensive and more research should be performed in this field. Future work in the development of component-based technologies could include determination of more quality metrics for components that are easy to calculate and more feasible to use. It is important to devise a formal methodology for determining the relative weights to be assigned to the different quality metrics based on stakeholder input. In this regard, a potential approach may be to use the analytic hierarchy process. In real-world COTS selection problem, the decision maker may generate suitable possibility distributions based on subjective judgments and/or historical data. Thus, it would be interesting to apply trapezoidal, bell-shaped, triangular or other possibility distribution patterns for representing imprecise numbers in solving COTS selection model using the fuzzy logic approach.

REFERENCES

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

[2]. Voas, J. and Agresti, W. W. 2004. Software quality from a behavioral perspective. IT Professional, 6,4 (July 2004), 46–50.

[3]. IEEE Standards Board (1990). “IEEE Standard Glossary of Software Engineering Terminology”, Computer Society of the IEEE.

[4]. G. Ruhe, "Intelligent Support for Selection of COTS Products,", LNCS, Springer, vol. 2593 2003. pp. 34-45

[5]. Mohamed, Abdallah Sami Abbas Shehata. Decision support for selecting COTS software products based on comprehensive mismatch handling. Diss. University of Calgary, 2007.

[6]. Pande, Jeetendra, Christopher J. Garcia, and Durgesh Pant. "Optimal component selection for component based software development using pliability metric." ACM SIGSOFT Software Engineering Notes 38.1 (2013): 1-6.

[7]. Kaur, A. and Mann, K.S. 2010. Component selection for component based software engineering. International Journal of Computer Applications, 2, 1(2010),109–1sssssss14.