Estimation and Enhancement of Component Reusability by Selecting Appropriate Component Selection Methods

Aditya Pratap Singh
A. K. G. Engineering College, Ghaziabad, U. P., INDIA,

Pradeep Tomar
School of ICT
Gautam Buddha University, Greater Noida, U.P. INDIA

Abstract- The reusability of software component directly affects the overall quality of the Component-Based Software (CBS). Reusability of a software component is measured as the effortlessness of reuse as perceived by the subjects reusing the component. Reusability estimation becomes a significant choice for measuring quality of CBS. The metrics for estimation of component reusability are needed to ensure quality by researchers of Component-Based Software Engineering (CBSE). The reusability estimation requires certain factors to measure. This paper discusses some quality attributes of software components which are used to estimate the component reusability. Paper also discusses their effect on reusability of CBS. Based on these quality attributes this paper presents and discusses component selection methods and their usage which may enhance the component reusability.

Keywords – Software reuse, software measurement, component-based software development, Commercial Off-The-Shelf.

I. INTRODUCTION

The software industry is facing fundamental challenge that is caused by a rapidly growing demand for quick and cost-effective development of large and complex software systems. To conquer this challenge, software community has moved towards the component based software engineering (CBSE). CBSE provides the methodological facilities that enable the easy assembly and upgrading of the software systems out of independently developed pieces of the software. The building of systems from components and the building of components for different systems requires established methodologies and processes not only in relation to the development/maintenance aspects, but also to the entire component and various aspects of system lifecycle (Crnkovic, 2002). According to (Clements, 1996), CBSE embodies the buy, don't build philosophy. The author also stated about CBSE that in the same way that early subroutines liberated the programmer from thinking about details, CBSE shifts the emphasis from programming to composing software systems. Effective reuse of software components helps in development of quality product, in reducing effort needed for testing and maintenance and in reducing cost. With such capabilities reusability became a potential choice among software developers of modern era. Reusability is the main feature of software component which differs from other software assemblies (Singh, 2012). 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 (Kumar, 2011). In case of components reuse may be a black box reuse in which the developer could only see the interface and not implementation. This makes the process of retrieval and selection of suitable software components from a large library of components very critical.

The paper is structured into four remaining sections: Section II reviews the Software Reusability and existing methods to enhance reusability. Section III presents the identified Quality factors affecting reusability. The discussion on available component/COTS selection methods and process of selection appropriate selection method is given in section IV and V. Conclusion is given in section VI.

II. LITERATURE REVIEW

According to Hopkins (Hopkins, 2000) reusability is an important engineering driver in the development of a Component-Based Software (CBS) and says that in the context of CBSE, reusability can refer to, the ability to reuse existing components to create a more complex system and according to Pressman (Pressman, 2010) reusability of software is the extent to which a program or part of a program can be reused in other applications related to packaging and scope of the functions that the program can perform. The software reusability is used to measure the degree of features that are reused in other applications. It is an attribute that refers to the expected reuse potential of a component (Gill, 2003). Reusability is not only meant to construct system from reusable existing components but it is also building components intended for reuse. Software reusability becomes a key to improve software development productivity and quality. The component reuse is a better approach if the cost of reuse is less than the cost of developing new components. The efforts needed to reuse a component in another system of the same domain may be determined on the basis of the scale provided in (Mao et al, 1998). Reusability can be measured only by studying its interface (Rotaru et al, 2005). Reusability can measure the degree of features that are reused in building applications. There is a number of metrics available for measuring the reusability for object-oriented systems. But there are some difficulties in applying existing object oriented metrics into the component development and CBSE (Sharma et al, 2007).

There is still a great thirst of reusability metrics for CBS. The requirement today is to relate the reusability attributes with the metrics and to find how these metrics collectively determine the reusability of the software component. Reusability metrics can be used to measure the software reusability in CBS with the help of selection of appropriate software component.

Washizaki (Washizaki et. al, 2003) proposed a metrics suite for measuring the reusability of black-box components for the activity of development with component reuse, based on the limited static information that can be obtained from the outside of components without any source codes, in order to identify the best components in terms of their reusability. Metrics are defined as ratios of the effective use of a given sort of interface features when compared to its potential use. Panwar and Tomar (Panwar and Tomar , 2011) proposed a method which finds the effects on reusability through changes in early phases of software developing process like requirements analysis, design, coding and simultaneously on reliability of code or program and it would also helpful to find the number of faults. Cho (Cho et al, 2001) has proposed a set of metrics named Component Reusability (CR) and Component Reusability Level (CLR). However proposed metrics are based on lines of codes and can only use at design time for components.
There are various factors considered by researchers for measuring reusability. According to Sharma et al. (Sharma, 2009) there are four factors namely customizability, portability, understandability and interface complexity of component for reusability assessment and used Fuzzy Logic based approach. Washizaki (Washizaki et. al, 2003) proposed the Component Reusability Model from the user's point of view and the application point of view. This model identified factors namely understandability, portability and adaptability. Rotaru (Rotaru et al, 2005) considered adaptability, complexity and compose-ability of a component to measure reusability. According to Sangwan et al. (Sangwan et. al., 2011) Changeability, Interface Complexity, Understandability of Software and Documentation Quality can be used as attributes for accessing software reusability. Authors proposed a soft computing technique to automatically predict software reusability based on above parameters. Gill (Gill, 2006) argues the importance of component characterization for its better reusability and also discusses several benefits of component characterization, which includes improved cataloguing, improved usage, improved retrieval and improved understanding eventually for better reuse.

III. QUALITY FACTORS AFFECTING REUSABILITY

On the basis of literature survey, this study finds following factors used by many researchers significant to estimate the software reusability.

3.1 Interface complexity

Component reuse is a black-box reuse, where the application developer can only access the interface, not the implementation of the component. The interface contains public methods, user documentation, requirements and restrictions of the component. These interface methods may have parameters and return values, which are the only source of information available to us. Depending on the nature and the number of these parameters and return values and properties, some weight values may be assigned to them, which can be used to measure the complexity of the target interface method. Higher interface complexity will lead to the high efforts for understanding and customizing the components. To enhance reusability the interface complexity should be as low as possible.

3.2 Customizability

Customizability is defined as the built-in capability for supporting to modify and configuration of a component's internal functional features as per the application requirement. Components must allow their attribute values to be changed programmatically or through some visual interface for their customization. Customizability will lead to better reusability in CBS and thus help in maintaining the component in the later phases. The number of writable properties provided by a component strongly affects the customizability of the component. Hard to customize component leads towards poor reusability.

3.3 Reliability

CBS reliability is estimated using the reliability of the individual components and their interconnection mechanisms (Singh and Tomar, 2013). The reliability study is conducted mainly depending on failure data, assuming a parametric model of the cumulative number of failures. To quantify the failure behavior of a software system, software reliability is an important measure to facilitate developers to arrange sufficient test activities. Components with higher reliability will lead to higher reusability.

3.4 Documentation Quality

Good quality documentation for components is required so that application developers, who are using these components, will have thorough understanding of component. The documentation of component refers to component manuals, demos, help information, and marketing information. Documentation should also describe the component's context dependencies and architectural constraints. A reusable component is a box with specified interface. The interface contains public methods, user documentation, requirements and restrictions of the component. Generally, good guidelines for predicting reusability are: small size of code, simple structure and good documentation (Sagar et al, 2010).

There may be some more factors which are selected for reusability measurement in different context like Singh and Tomar (Singh and Tomar 2016)

IV. COMPONENT/COTS SELECTION METHOD

One of the important processes of CBSE is the proper selection of the components. The Component selection comprises following activities (Carvallo et al, 2007):

- Acquiring and specifying the requirements,
- Understanding the available components,
- Assessing component compatibility with regard to the requirements, and
- Selecting the "best" available component.

Conduct of unstructured searches is probably the most common way of finding components if team members are less experienced. Components may be identified through search engines like Google, general software repositories like SourceForge and language or domain specific repositories and sites like CPAN, The ServerSide and so on (Hauge et al, 2009). General COTS selection process (GCS) (Mohamed et al, 2007) an abstract procedure with the steps: Define criteria, Search for products, Create shortlist, Evaluate candidates, Analyze data and select product., has a few properties which are worth noticing; the requirements are expected to be defined and weighted up front, it is a best fit formal or mathematical competition between several likely candidate components, and the identification of these components is basically ignored. The components are described by their functionality offered and need to be selected depending on their individual cost and on the context of the CBS in which they are employed. In general, there may be different alternative components are available and need to be selected, each coming at their own cost.

Some direct assessment methods proposed for COTS selection are the Off-The-Shelf-Option (OTSO) (Kontio, 1996), COTS-based Integrated System Development (CISD) (Tran et al, 1997), Procurement-Oriented Requirements Engineering (PORE) (Maiden and Ncube, 1998), Social-Technical Approach to COTS Evaluation (STACE) Framework (Kunda and Brooks, 1999) and COTS-based Requirements Engineering (CRE) (Alves and Castro, 2001), and the. So far, most of the Selection process concentrated on the functional aspects of components, leaving aside the (difficult) treatment of their quality and extra-functional properties.

Off-The-Shelf Option (OTSO) method that provides specific techniques for defining evaluation criteria, comparing the costs and benefits of alternative products, and consolidating the evaluation results for decision-making. The definition of hierarchical evaluation criteria is the core task in this method, it identifies four different sub processes: search criteria, definition of the baseline, detailed evaluation criteria definition, weighting of criteria. Even though OTSO realizes that the key problem in COTS selection is the lack of attention to requirements, the method does not provide or suggest any specific solution. The method assumes that requirements already exist since it is based on the requirements specification for defining the evaluation criteria. Analytic Hierarchy Process (AHP) is used to consolidate the evaluation results for decision-making.

COTS-Based Integrated System Development (CISD) Method has a two-stage selection process. The first stage is product identification, in which candidate components are identified and classified. The second is evaluation, in which the final candidate components are chosen (and unsuitable candidates removed). This method is useful when multiple homogeneous COTS products were required.

Procurement-Oriented Requirements Engineering (PORE) a template-based method for requirements acquisition integrates techniques and product selection with process guidance for choosing and using each technique. It draws on techniques from several different disciplines, including knowledge engineering techniques such as card sorting and laddering, Feature analysis techniques to aid when scoring the compliance of each product to each requirement, Multicriteria decision making (MCDM) techniques and outranking methods to aid decision making during the complex product ranking and selection process and Design rationale techniques to record and aid this decision-making process. The COTS selection is made by rejection.

Socio-Technical Approach to COTS Evaluation (STACE) considers the whole COTS evaluation and selection process and emphasizes customer participation during evaluation. This method studies how to define evaluation criteria. There are three underlying philosophy of STACE: (i) Support for a systematic approach to COTS evaluation and selection, (ii) Support for both evaluation of COTS products and the underlying technology, and (iii) Use of socio-technical techniques to improve the COST selection process.

COTS-based Requirements Engineering (CRE) Method was proposed to facilitate a systematic, repeatable and requirements-driven COTS selection process. A key issue supported by this method is the definition and analysis of non-functional requirements during the phases of COTS evaluation and selection. The selection of COTS products is made by rejection. The method has four iterative phases: Identification, Description, Evaluation, and Acceptance. Unlikely other approaches that analyze only functional requirements to evaluate COTS, this method also cover the non-functional characteristics of the candidate component. Decision making analysis is performed using Multiple Criteria Decision Making (MCDM) technique.

V. DISCUSSION AND FINDINGS

The Component selection process is a decisive activity of COTS-based development when the quality and suitability of the product have to be verified with respect to organization requirements and business expectations. To facilitate knowledge reuse in such a scenario, tools supporting the dynamic selection of reusable knowledge components are crucial. The above discussed techniques handle mismatches and other factors partially. The main shortcomings of the discussed methods for COTS selection are:

- In broad sense, these methods rely on the definition of pre-established and structured criteria based on set requirements. These approaches are not appropriate to handle with the impositions of a highly unstable and uncertain COTS marketplace.
- These methods lay emphasis on the importance of requirements analyses in order to conduct a successful selection. Although none of them support the complex process of requirements analysis and balancing with COTS features limitations.

To select best evaluation technique for COTS selection the issues like COTS Integration, Mismatch of non-functional requirement, Indecision Handling (completeness, accuracy, and consistency and Rotating demands of Stakeholders are to be handled. Different COTS evaluation methods are proposed for different domains. There is no such method available that give solution for all above issues. To date, Component selection techniques often make an ideal assumption that there is one-to-one mapping between requirements and components. In reality, components are usually designed for general purposes and provide a range of features that can be adapted to meet the needs of a CBS. This implies that a component can potentially match more than one functionality. On the other hand, system requirements are usually not independent of each other and a component selection process needs to consider the dependencies between system goals. Some of the available methods (e.g., PORE, CEP, STACE) are process oriented and offer (intelligent) support for negotiation to determine a 'good' fit between evolving requirements, candidate COTS, and architectures. The current COTS selection methods not at all tackle uncertainty. None of these selection methods judge vagueness in an inclusive style. Only one selection method i.e. PORE holds a prototype tool. There is no software tool supporting other COTS selection methods negatively pressures their usability.

This paper also identifies selection policies that guide the integration of components in applications and motivate the suitability of CBS to implement a flexible component selection and monitoring mechanism.

5.1 Identifying component selection policies

In order to accomplish active component selection based on business requirements, the proposed system need to define, as part of their requests, not only which functionality is needed but also which non-functional requirements (cost, resource required, reliability and dependability) are pursued. The component selection process should select the component that best fit the specified requirements and swap to them when needed. A component complies with a selection policy if it satisfies its condition. Only if a component complies with all specified selection policies it is approved to be considered during the selection process. An approved component can be selected and integrated in the system. If a component does not satisfy at least one selection policy, then it is disapproved and rejected for selection. Component selection policies can be classified as necessary and guidelines. A necessary selection policy imposes a constraint on a component that should be satisfied at the moment the component is invoked. The constraint can contain absolute conditions, if two or more component satisfies a necessary policy it is not determined which one must be chosen. This is where guidelines policies come in. A guideline specifies that if multiple components are available to provide the required functionality, one is preferred over the other.

5.2 Towards a flexible implementation of selection policies

After identifying selection policies, the challenge is how to implement them in a flexible way, avoiding the applications to change each time the business requirements change. For example sometime the requirements changes from cheapest component to component with higher reliability. The selection process being used needs to constantly cope with changes in the business requirements embodied in their policies. The selection process must be flexible enough to deal with the changed selection criteria. In the context of CBSE, it should be possible to integrate the identified component quality factors and selection policies that best accommodate to the application requirements. Selection policies should be kept separated from the components and applications that integrate them to enhance maintainability, reusability and adaptability.

VI.CONCLUSION

The improved reusability not only improves productivity but also has a great impact on quality of the product. This paper highlights the relevant quality attributes related to reusability of CBS, which plays significant role in estimation of reusability. A number of selection processes that support COTS software components selection were discussed. The study aims at a selection of appropriate component/COTS selection method that guarantees the optimality of the generated component system. In the presence of huge component repositories, the best solution is not trivial to obtain. The support in the form of findings is presented to identify the selection method for selection of reusable components based on identified quality attributes by using proposed selection policies to enhance reusability.

REFERENCES

[1] Alves C. and Castro J. (2001), “CRE: A Systematic Method for COTS Selection”. In Proceedings of XV Brazilian Symposium on Software Engineering, Brazil, Vol. 05 pp. 473-493.

[2] Carvallo, J. P., Franch, X., and Quer, C. (2007), “Determining Criteria for Selecting Software Components: Lessons Learned”. IEEE Trans. On Software, Vol. 24, No 3, pp. 84-94.

[3] Cho, E. S., Kim, M. S., Kim, S. D., (2001), “Component Metrics to Measure Component Quality”, 8th Asia-Pacific Software Engineering Conference, Macau, pp. 419-426.

[4] Clements, P. C. (1995), “From Subroutines to Subsystems: Component-Based Software Development”. American Programmer, Vol. 8, pp. 31-31.

[5] Crnkovic, I. (2002), “Component□Based Software Engineering-New Challenges in Software Development”. Software Focus, Vol. 2, No. 4, pp. 127-133.

[6] Gill, N. S., (2003), “Reusability Issues in Component-Based Development”, ACM SIGSOFT Software Engineering Notes Vol. 28 No. 6, pp. 30.

[7] Gill, N. S. (2006), “Importance of Software Component Characterization for Better Software Reusability.” ACM SIGSOFT Software Engineering Notes, Vol. 31 No. 1, pp. 1-3.

[8] Hauge, O., Osterlie T., Sorensen C. F., and Gerea M. (2009), “An Empirical Study on Selection of Open Source Software - Preliminary Results”. Proceedings of IEEE Computer Society ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, FLOSS '09. Vancouver, Canada, pp. 42–47.

[9] Kontio J. (1996), “A Case Study in Applying a Systematic Method for COTS Selection”. In Proceedings of ICSE-18, pp. 201–209.

[10] Kumar, A., Tomar, P., and Panwar, D. (2011), “Challenges Faced During Software Component Selection”. International Journal of Computer Science and Engineering. Vol. 1, No. 1, pp. 10-14.

[11] Kunda, D. and Brooks, L. (1999), “Applying Social-Technical Approach for COTS Selection”. Proceedings of the 4th UKAIS Conference, University of York, pp. 552-565.

[12] Maiden N. and Ncube C. (1998), “Acquiring COTS Software Selection Requirements”. IEEE Software, pp. 46–56.

[13] Mao Y., Sahraui H.A., and Lounis H. (1998), “Reusability Hypothesis Verification Using Machine Learning Techniques: A Case Study”. In Proceedings of the 13th IEEE International Conference on Automated Software Engineering, pp. 84-93.

[14] Mohamed, A., Ruhe, G., Eberlein, A. (2007), “COTS Selection: Past, Present, and Future”. In Proceedings of the 14th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS’07), pp. 103–114.

[15] Ncube C. and Dean J.C., (2002), “The Limitations of Current Decision-Making Techniques in the Procurement of COTS Software Products”. Proceedings of ICCBSS, Orlando, Florida USA, pp. 176–187.

[16] Panwar, D., Tomar, P. (2011), "New Method to Find the Maximum Number of Faults by Analyzing Reliability and Reusability in Component-Based Software", 3rd IEEE International Conference on Trendz in Information Sciences and Computing (TISC), pp. 164-168.

[17] Rotaru O P, Dobre M, Petrescu M. (2005), “Reusability Metrics for Software components”, Proceeding of the 3rd ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-05), Cairo, Egypt, pp. 24-29.

[18] Sagar, S. and Nerurkar, NW and Sharma, A. (2010), “A Soft Computing Based Approach to Estimate Reusability of Software Components”, ACM SIGSOFT Software Engineering Notes, Vol. 35, No. 5, pp. 1-5.

[19] Sharma, A. and Kumar, R. and Grover, P.S. (2007), "A Critical Survey of Reusability Aspects for Component-Based Systems", Proceedings of the World Academy of Science, Engineering and Technology, Vol. 21, pp. 35-39.

[20] Sharma, A., Grover, P. S., and Kumar, R. (2009), “Reusability Assessment for Software Components”. ACM SIGSOFT Software Engineering Notes, Vol. 34, No. 2, pp. 1-6.

[21] Singh, A. P., Kumar E., Tomar P. (2012), “ A Classified Survey of Component Based Software Engineering”, proceeding of 1st International Conference on Innovations and Advancements in Information and Communication Technology, ICIAICT’12 , Gautam Buddha University, Greater Noida, India, Vol. 1, pp. 128-136.

[22] Singh, A. P., Kumar E., Tomar P. (2013), “A New model for Reliability Estimation of Component-Based Software”. 3rd IEEE International Advance Computing Conference (IACC-2013), Ghaziabad, India.

[23] Singh, Y. and Bhatia, P.K. and Sangwan, O. (2011), “Software Reusability Assessment using Soft Computing Techniques”, ACM SIGSOFT Software Engineering Notes, Vol. 36, No. 1, pp. 1-7.

[24] Tran, V., Liu, D. B., and Hummel, B. (1997), “Component-Based Systems Development: Challenges and Lessons Learned”. In Proceedings of Eighth IEEE International Workshop on Software Technology and Engineering Practice. pp. 452-462.

[25] Washizaki, H., Yamamoto, H., and Fukazawa, Y. (2003), “A Metrics Suite for Measuring Reusability of Software Components.” In Proceedings of IEEE Ninth International Software Metrics Symposium, pp. 211-223.

[26] Singh, A. P., and Tomar, P. (2016). “Web Service Component Reusability Evaluation: A Fuzzy Multi-Criteria Approach”. International Journal of Information Technology and Computer Science (IJITCS), Vol. 8, No. 1, pp. 40-47.