***I.J. Engineering and Manufacturing*, 2014, 1, 18-24**
Published Online May 2014 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijem.2014.01.03

Available online at http://www.mecs-press.net/ijem

Improved Framework for Appropriate Components Selection

Weam Gaoud Alghabbana[a], M. Rizwan Jameel Qureshi [a,*]

[a]*Faculty of Computing and Information Technology, King Abdilaziz University, Jeddah, Saudi Arabia*

Abstract

Component based software engineering (CBSE) approach is used to develop a software system from pre-existing software components. It is difficult to select a suitable component from the library of components for the problem of CBSE. Appropriate software component selection plays an important role for the success of project. Many approaches are suggested to solve component selection problem. In this paper, the component selection is done by improving the integrated component selection framework by including the pliability metric to cut down the time and cost. Pliability is a flexible measure that assesses software quality. We anticipate that the proposed solution will solve the component selection problem of software industry.

Index Terms: Component Based Software Engineering (CBSE), pliability, framework.

1. Introduction

Component Based Software Engineering (CBSE) is concerned with selecting and designing components. It is centered on the idea of developing a system from pre-existing components. Designing a system by reusing existing components leads to reduce time to market. However, it is always difficult for a software company to select appropriate set of components as per requirements of each project. There is a need to include enough information to facilitate system architect while selecting components during analysis phase.

Several papers are written to address the component selection problem. Chad et al., 2011 describe that one of the solutions is integrated framework. The integrated framework helps to select components using system modeling and statistical analysis. Jeetendra et al., 2013 produced the pliability metric that measures software quality as a function of component's quality.

This paper is organized further as follows. Section 2 provides an overview of the literature. Problem definition and the proposed solution of the problem are illustrated in section 3.

* Corresponding author. Tel.: +966536474921; fax: +966-6952548
E-mail address: anriz@hotmail.com

2. Related Work

According to Lycett and Giaglis, 2001, it is extremely difficult to evaluate information systems in terms of reuse. They are of the view that all development approaches have a danger and risk to reuse the existing components.

The main reuse risks can be avoided as following by Lycett and Giaglis, 2001.

- Integrate business driven evaluation at an early stage during selection of components. It will reduce effort of evaluation and selection.
- Evaluation and selection should be an ongoing process.

According to the de Jonge, 2003, one of the reuse requirements is to develop independent components and these must be integrate-able. He proposed techniques to integrate reusable independent components within one system and across systems. The proposed source tree composition technique integrates;

- core modules of components;
- package development that permits development of more than one component concurrently according to the scope of different software systems.

Package based software development is a popular research area. The author concentrates on source tree composition technique for software configuration management of more than one system. This is a problem in terms of management of reusable components in a repository. The author did not discuss crosscutting development of components and their integration in multiple systems.

The extent of software reuse depends upon the reuse strategy followed by Marcus et al., 2003. Marcus et al., 2003 proposed a set of six dimensions to support the reuse practices after conducting a survey during which data was gathered from 71 software development groups. These dimensions were planning and improvement, formalized process, management support, project similarity, object technology and common architecture. On the basis of dimensions, they discovered five reuse strategies that are practiced in software development groups. The five strategies were ad-hoc reuse with high reuse potential, uncoordinated reuse attempt with low reuse potential, uncoordinated reuse attempt with high reuse potential, systematic reuse with low management support and systematic reuse with high management support. Main objective of the research by Marcus et al., 2003 is to classify reuse strategies so that development groups can get benefit and achieve success to complete software projects. The authors support the last strategy i.e., systematic reuse with high management support, but this scheme needs highly detailed analysis. This analysis consists of gathering of data from various software houses working in different geographical locations, before reaching a conclusion.

Selection of reusable components is important to improve productivity of component-based software by Jihyun et al., 2003. This research proposed a Component Repository for facilitating Enterprise Java Beans (EJB) Component Reuse (CRECOR) to store and manages the reusable components. Working on the reusable components with repository (software) has many benefits for example, specification viewing, adapting, testing and deploying. The repository proposed by Jihyun et al., 2003 does not have a version control and change control functions to manage different versions of reusable components. Version and change control are important functions of a repository to manage and update different versions of a component.

Table 1. Summary of related work

Paper Title	Summary
A Metrics-Based Evolutionary Approach for the Component Selection Problem. Vescan, 2009.	• A metric is used to select component based on three attributes: cost, reusability and functionality. • Compatibility issues between two connected components are not discussed.
Software Component Retrieval Using Genetic Algorithms. Dixit and Saxena, 2009.	• Component selection process is time and cost consuming.
Optimization of Software Components Selection for Component-Based Software System Development. Kwong et al., 2010.	• Software development team use functional characterization to select suitable component as per requirements.
A Hybrid Evolutionary Multiobjective Approach for the Dynamic Component Selection Problem. Vescan et al., 2011.	• Hierarchies of components are not considered.
Component Selection for Component based Software Engineering. Kaur and Mann, 2010.	• Case studies are conducted on small projects to validate and generalize the results.
An Optimization model for Software Component Selection under Multiple Applications Development. Tang et al., 2011.	• Customized genetic algorithm does not guarantee the optimal solution.
Algorithm for Component Selection to Develop Component-Based Software with X Model. Tomar and Gill, 2013.	• Time and cost of development are high.
Optimal Component Selection for Component Based Software Development using Pliability Metric. Jeetendra et al., 2013.	• The proposed model does not include quality metrics to select a suitable component.

Haddad, 2006 is of the view that software organizations have to invest huge sums of money to start successful reuse methodology and it's a barrier for them. The author believes that core of reuse is source code. According to an estimate mentioned by the author, "domain specific components represent up to 65% of the application size. One approach for effective reuse practices focuses on domain specific components". The author proposed an integrated approach for component-based development to support domain specific components. An integrated approach is a collection of reusable components in a development environment. The author also discusses the concept of interface to describe a wrapper interface mechanism. The wrapper interface mechanism can be used to manage and control the interface between or among integrated collection of reusable components. The objective of this research is to develop benchmarks for software organizations so that these begin reuse practices by emphasizing mainly on programming effort and not on management and operational issues. Focus of Haddad's research is development of domain specific reusable components and not construction of reusable components of different domains of concern. This problem can be handled through software engineering for adaptive and self-managing systems. Michael et al., 2006 proposed few recommendations for component selection without validation.

The research by Arndt and Dibbern, 2006 reveals that there are two traditional approaches to construct software systems i.e., customization and use of standard components libraries. A composition of customized and component libraries domains approaches has been proposed to achieve benefits of both. There are many factors to resist practice of this new domain. A change process is explained to support composition approach by providing a logical solution. A concept is also introduced for mindful innovation to show that how modular development can avail the benefits of domain change. The advantages of Component Based Software Development (CBSD) domain are also discussed that are already described in many papers. This work explains this concept theoretically. However, this research is conducted to support the practice of CBSD domain. CBSD acceptance process is not considered by Arndt and Dibbern, 2006.

Khemakhem et al., 2006 have presented SEC (search engine for component-based development). SEC helps developers to identify and assemble components using a repository. It does not use ontology-based description technique to retrieve components. An advanced version of SEC is presented by the authors in

Khemakhem et al., 2007. This version uses ontology-based technique to identify and retrieve components. The searching engine is deficient in wrapping complete set of ontologies. SEC has not been tested for a variety of CASE tools. Jiang Guo, 2006 discusses various integration issues in the current ERP systems. The author suggests that most of the integration issues are resolved by using category theory. This research objective is to propose a framework for component dependencies modeling technique. The proposed framework requires further evaluation through case studies.

The proposed model by Washizaki et al., 2006 integrates components using method type collection. It facilitates software engineering team in component qualification by using signatures. The connection model is based on Java and it does not support other architectures such as Microsoft. The concept needs further validation for large and complex applications. A comparative analysis of the existing process models has been presented by Crnkovic et al., 2006. The objective is to describe different phases of component-based system life cycle. The authors have partitioned component-based development processes into system development and component development. They have not introduced any new concept. CBSE is already divided into two domains which are CBD and domain engineering by Pressman, 2005.

The existing components selection methods do not address specification of functional and non-functional requirements. Component selection decision is important because the integration risks can be solved by the right selection of components. Dixit and Saxena, 2009 proposed genetic algorithms based approach to solve the problem of component selection. Tang et al., 2011 proposed an optimization model to solve the problem of component selection including reusability and compatibility at the same time. This model depends on genetic algorithm and that helps the developers to select components when they are working on multiple applications concurrently. Designing a system by using the existing components can reduces the time to market. Chad et al., 2011 proposed a framework for components selection using simulated annealing algorithm. CBSE includes reuse of components to enhance software quality by improving functionality, security, cost and maintainability. Jeetendra et al., 2013 developed a flexible evaluation model to enable optimal component selection based on different quality metrics of component and cost. Table 1 illustrates the summary of related work.

3. Problem Statement and the proposed solution

Each component in the library satisfies a group of requirements. The research question therefore becomes:

"How to find a set of suitable components that satisfy the requirements?"

It is a matter of great challenge to select a set of suitable components from library to satisfy functional and non-functional requirements. There are several approaches to address this challenge like Chad et al., 2011 approach. The proposed framework uses system modeling and analysis that is the first phase as shown in fig. 1. The regression analysis computes the dependency of system performance with attributes of individual components. The probability analysis computes how much a component matches with the requirements. The outputs of this phase are regression equations and conditional probabilities those are used in the second phase.

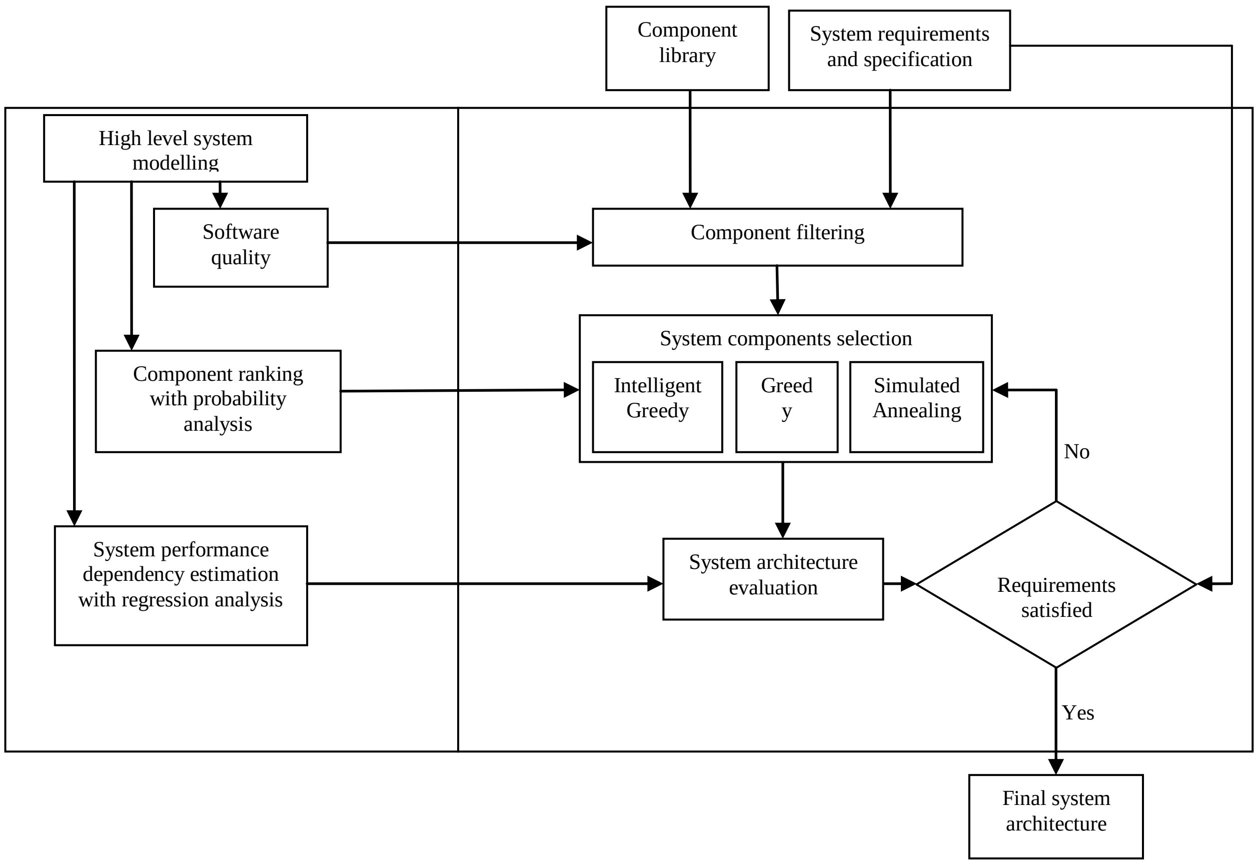
Component library
System requirements and specification
High level system modelling
Software quality
Component filtering
System components selection
Intelligent Greedy
Greed y
Simulated Annealing
Component ranking with probability analysis
No
System performance dependency estimation with regression analysis
System architecture evaluation
Requirements satisfied
Yes
Final system architecture

Fig. 1.The proposed integrated component selection framework

The second phase begins with the system requirements and specification. The search space is reduced by eliminating the components those do not satisfy the system specification constraints. The selection algorithm is chosen from greedy, intelligent greedy and simulated annealing algorithms. Each algorithm produces a series of components and performances of components are estimated using regression equations to match with requirements. Pliability metric is used to evaluate software quality. Jeetendra et al., 2013 introduce the pliability metric to measure software quality as a function of individual components. This research proposes an approach to select suitable component from a library of components. This is achieved by integrating the pliability metric and the existing integrated component selection framework. The total software quality measure ‘Q’ are defined using set of quality attributes such as reliability, performance, fault tolerance, safety, security, availability, testability and maintainability. Software quality is expressed as weighted linear combination of values in eq. (1).

$$Q= wRR + wPP+ wFF+ wSaSa+ wSeSe+ wAvAv+wTT+ wMM \qquad (1)$$

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

A weighted value is assigned to the selected attributes and the sum of all weighted values is equal to 1 as defined in eq. (2).

$$wR + wP + wF + wSa + wSe + wAv + wT + wM = 1 \tag{2}$$

This metric provides a flexible way to assign weighted values to all attributes. A normalized value between ‘0’ to ‘10’ is calculated using formula in eq. (3).

$$qhi = \left((Raw\ QA\ (h,i))/(Max\ (h)) \right) \times 10 \tag{3}$$

4. Conclusion

Software development companies face it difficult to select suitable components that match with requirements of new project. Studies show that there is a need to deal with the component selection problem to meet the industrial demand. On the face of it, a novel solution is proposed by integrating pliability metric and component selection framework. The proposed solution uses pliability metric to include quality attributes while selecting components. This proposed solution will help the system architect during analysis phase to select appropriate components to save time and cost.

References

[1] Anurag Dixit, P.C. Saxena. Software Component Retrieval Using Genetic Algorithms. in Proc. International Conference on Computer and Automation Engineering, IEEE, 2009;151-155.
[2] Chad Calvert, Georgiana Hamza-Lup, Ankur Agarwal, Bassem Alhalabi. An Integrated Component Selection Framework for System-Level Design. in Proc. 4th Annu. IEEE Int. Syst. Conf., 2011; 261–266.
[3] Crnkovic, I. Chaudron, M. and Larsson, S. (2006). Component-Based Development Process and Component Lifecycle. Proc. Int. Conf. Software Engineering Advances ICSEA'06, Tahiti, French Polynesia, 44.
[4] De Jonge, M. (2003). Package-Based Software Development. Proc. 29th EUROMICRO Conference on New Waves in System Architecture, Belek-Antalya, Turkey, 76-78.
[5] Haddad, H. M. (2006). Integrated Collections: Approach to Software Component Reuse. 3rd International Conference on Information Technology: New Generations, Las Vegas, Nevada, 28-33.
[6] Jeetendra Pande, Christopher Garcia, Durgesh Pant. Optimal Component Selection for Component Based Software Development using Pliability Metric. ACM SIGSOFT Software Engineering Notes 2013; 38:1-6.
[7] Jihyun, L. Jinsam, K. Gyu-Sang, S. (2003). Facilitating Reuse of Software Components using Repository chnology. Proc. 10th Asia-Pacific Conf. Software Engineering, Chiang Mai, Thailand, 136-142.
[8] Khemakhem, S. Drira, K. and Jmaiel, M. (2006). SEC: A Search Engine for Component Based Software Development. Proc. ACM symposium on Applied computing, Dijon, France, 1745-1750.
[9] Khemakhem, S. Drira, K. and Jmaiel, M. SEC+: An Enhanced Search Engine for Component-Based Software Development. ACM SIGSOFT Software Engineering Notes 2007; 32, 1-6.
[10] Kwong C.K., Mu L.F., Tang J.F., Luo X.G. Optimization of Software Components Selection for Component-Based Software System Development. Journal of Computers and Industrial Engineering 2010;58:618-624.
[11] Lycett, M. and Giaglis, G. M. (2001). Component Based Information Systems: Towards a Framework for Evaluation. Proc. 33rd Annual Int. Conf. System Sciences, Hawaii, 10.

[12] Marcus, A. R. Keven, J. D. Uday, R. K. and Nader, N. Strategies for Software Reuse: A Principal Component Analysis of Reuse Practices. IEEE Transactions on Software Engineering 2003; 29:825-837.
[13] Tang J. F., Mu L.F., Kwong C.K., Luo X.G. An Optimization model for Software Component Selection under Multiple Applications Development. European Journal of Operational Research, 2011; 301–311.
[14] Tomar P., Gill N. New Algorithm for Component Selection to Develop Component-Based Software with X Model. Lecture notes on Software Engineering, Vol. 1, No. 3, Aug 2013; 298-302.
[15] Vescan A., Gros C., Yang S. A Hybrid Evolutionary Multiobjective Approach for the Dynamic Component Selection Problem. in Proc. 11th International Conference on Hybrid Intelligent Systems (HIS), 2011; 714-721.
[16] Vescan A. A Metrics-Based Evolutionary Approach for the Component Selection Problem. in Proc. Computer Modelling and Simulation, 2009; 83-88.
[17] Washizaki H., Hoshi D., Yoshiaki F. A Flexible Connection Model for Software Components. IEICE-Transactions on Information and Systems 2006; 4:1421-1431.

Authors' Profile

Dr. M. Rizwan Jameel Qureshi received his Ph.D. Computer Sciences degree from National College of Business Administration & Economics, Pakistan 2009. He is the best researcher awardees of King Abdulaziz University, Jeddah, Saudi Arabia in 2013 and Department of Computer Science, COMSATS Institute of Information Technology, Lahore, Pakistan in 2008.

Weam Gaoud Alghabban is a graduate Student of Information Technology, Faculty of Computing & Information Technology, King Abdulaziz University.

How to cite this paper: Weam Gaoud Alghabbana, M. Rizwan Jameel Qureshi,"Improved Framework for Appropriate Components Selection", IJEM, vol.4, no.1, pp.18-24, 2014.DOI: 10.5815/ijem.2014.01.03