Towards Component Evaluation Approaches

Suryani Ismail
Computer Science Department
Universiti Malaysia Terengganu (UMT)
21030 Mengabang Telipot,
Kuala Terengganu
sue@umt.edu.my

Wan M.N. Wan-Kadir
Software Engineering Department,
Fac. of Computer Science & Info. Systems,
Universiti Teknologi Malaysia,
81310 UTM Skudai, Johor, Malaysia
wnasir@utm.my,

Yazid M. Saman
Computer Science Department
Universiti Malaysia Terengganu (UMT)
21030 Mengabang Telipot,
Kuala Terengganu
yazid@umt.edu.my

Siti Zaiton Mohd Hashim
Software Engineering Department,
Fac. of Computer Science & Info. Systems,
Universiti Teknologi Malaysia,
81310 UTM Skudai, Johor, Malaysia
sitizaiton@utm.my

Abstract

Component-based software engineering (CBSE) is becoming popular due to the benefits of software reuse and the availability of many alternatives of reusable components. By the reuse of existing software components that have already been tested, software engineers may reduce errors and shorten the time to market of the system under development. However, software engineers still have a problem in selecting the existing reusable components as well as difficulties in determining the quality of the developed components for future reuse. Therefore, an appropriate component evaluation approach is extremely needed to harvest the full benefits of software reuse. In this paper, we present the comparative evaluation of the state-of-the-art component evaluation approaches with an emphasize on reusability. The results presented in this paper are very useful in achieving the ultimate aim of our research i.e. providing a holistic component evaluation approach that may assist software engineer both in the selection of the existing reusable components and the development of new reusable components.

Keywords: Component-based Software Engineering (CBSE), Component Reuse, Component Evaluation, Reusability Metrics, Component Quality.

1. Introduction

Component-based software engineering (CBSE) is becoming a popular software process model due to the benefits of software reuse and the availability of many alternatives of reusable components.

There are many benefits of software reuse in solving the inherent software engineering problems. It has been claimed to improve the productivity and the quality of software development [1-3]. Many organizations have benefited from using reusable components in reducing the time and cost of software development as well as improving the quality of the produced software [4-6]. Since software reuse is widely accepted as a solution in increasing the quality of both software product and process, there are many research efforts devoted to this area. For examples, the management of reusable components, reuse metrics, and the composition of reusable artifacts.

Apart from the benefit of software reuse, the availability of the reusable components also promotes the use of CBSE as a process model. Many developers develop reusable software components for their own future use or to be marketed as the common-of-the-shelf (COTS) components.

However, due to less effort given to the study of a component evaluation approach, software engineers still face the difficulties in developing and measuring the reusable components [7]. By the availability of such approach, it is easier for software engineers to determine the suitable components for their software project and verify the developed reusable components. In this paper, we aim at evaluating the state-of-the-art of component evaluation approaches and set the research direction in this crucial area.
This paper is organized as the following. Section 2 presents an overview of CBSE. Section 3 provides a brief description on the prominent component evaluation approaches. Section 4 explains briefly each characteristic of component evaluation approaches that mention in section 3. Section 5 describes the comparison criteria and the results of the comparison of the related approaches. Section 5 discusses the summary of the comparison results. Finally, its presents the conclusion and further work of this study.

2. Component-Based Software Engineering (CBSE)

CBSE is a software process model that emphasizes the design and construction of computer-based systems using reusable software artifacts such as architecture, design, framework, or components [8]. It aims at maximizing the reuse of existing software artifacts. The software units that are reused may be radically of different sizes, for example application system reuse, component reuse and functions reuse. However, although there has been interest in component reuse since the early 1980, it is only in the past few years that it has become accepted as a practical approach to software development [9]. Nowadays, due to the availability of many alternatives of common-off-the-shelf (COTS) software components, CBSE is becoming a popular software process model.

In general, CBSE is categorized into two main perspectives: development of components for reuse, and development of software systems using the reusable components [10]. In both perspectives, the component evaluation is considered as an important activity. The evaluation of the developed components may ensure their high degree of reusability whilst the evaluation of the several candidates of potentially reusable components may assist software engineers in choosing the appropriate component to be used in their software project.

Software component is defined as an implementation abstraction that is developed and packaged based on the aim of reuse, and it differs from code fragments, modules and programs [11]. It is also considered as a direct reuse of the software [12, 13], that can be deployed with little or no modification [14]. The idea of componentizing software had been suggested as a way of tackling the software crisis especially the late system delivery [15].

Currently, there is an increasing interest in CBSE research. In general, these researches can be grouped into seven categories i.e. component modeling and specification, retrieval techniques and specification matching, generative approach to component development, adaptation techniques, coordination and composition languages, verification, testing and certification, and configuration management [16].

3. Overview of Component Evaluation Approaches

Component evaluation is essential in assisting the selection of reusable components in a software development as well as determining the quality of the developed reusable components. Due to this reason, there are many approaches introduced for the evaluation of software components. Based on the exhausted review on the state-of-the-art approaches related to component evaluation, we categorize the approaches into four main groups i.e. Product Line Component, Original Component, Quality Component, and Reusable Component. It is found that the evaluation is primarily focused on the characteristics of the capability to support software component reuse. The characteristics considered by each approach are summarized in Table 1.

<table>
<thead>
<tr>
<th>EVALUATION APPROACHES</th>
<th>CHARACTERISTICS</th>
</tr>
</thead>
</table>
| Product Line Engineering Components (PLC) | • Understandability  
| | • Component replacetability  
| | • Functional commonality  
| | • Applicability  
| | • Non functional commonability  
| | • Variability richness  
| | • Tailorability |
| Originalty components (OC) | • Functionality  
| | • Reliability  
| | • Usability  
| | • Efficiency  
| | • Maintainability |
| Quality Components (QC) | • Functionality  
| | • Reliability  
| | • Usability  
| | • Efficiency  
| | • Maintainability  
| | • Portability |
| Reusable Components (RC) | • Portability  
| | • Adaptability (flexibility)  
| | • Understandability  
| | • Confidence (probability) |

3.1 Product Line Component (PLC)

Product Line Component (PLC) is a recent and affective approach to software component reuse [16]. Her et al. proposed that PLC consists of 7 evaluation characteristics. Two of them i.e. understandability and component replaceability are adopted from ISO 9126 whilst the remaining characteristics are newly defined as the core attributes for product line which include functional commonality, applicability, non functional commonality,
variability richness, tailorability [17]. Each characteristic is further described by its own sub-characteristics.

According to Clements, PLC consists of two processes namely core asset development and application engineering [16]. The success of the whole product line is determined by the reusability of a core asset [17]. The goal of this approach is to evaluate the reusability of core asset in Product Line Engineering.

3.2 Originals Components (OC)

Original components (OC) have smaller size and narrower functional scope and they usually find more uses in specific and dedicated functions.

OC has been proposed mainly based on the ISO 9126 quality model. The only major difference is the removal of the portability characteristic from the original ISO 9126 characteristics [7]. Apart from ISO 9126 quality model, it also adopted other models namely McCall’s model and Boehm’s model. The characteristics that are used in OC are functionality, reliability, usability, efficiency and maintainability. Similar to PLC, each characteristic has its own sub-characteristics. However, the sub-characteristics are modified and refined in the context of original components.

The main purpose of the OC approach is to evaluate the quality of the original components that will be used in the development process.

3.3 Quality Component (QC)

One of the major problems in CBSE is related to the quality of the components [18-20]. Without an acceptable standard of the quality level, the component usage may have catastrophic results [21]. According to Heinman and Council, the component quality problems must be resolved to increase the reliability and third-party certification programs that will increase the trust of the market oriented components [14]. Hence, the Quality Component (QC) approach is proposed in order to solve this problem.

As a result, researchers propose QC approach with consistent and well defined characteristics such as functionality, reliability, usability, efficiency, maintainability and portability for the component evaluation [18, 22, 23]. These characteristics have their own sub-characteristics.

The purpose of the QC approach is to evaluate and certificate software components that could determine their quality level.

3.4 Reusable Component (RC)

Reusable Component (RC) refers to the component evaluation approaches that emphasize specifically on component reusability.

Based on the McCall, ISO9126 and REBOOT, RC approach has been proposed which it can be used in evaluation reusable components. This approach includes four characteristics i.e. understanding, adaptability/flexibility, portability, and confidence/probability [24, 25]. Every characteristic has its own sub-characteristics.

The main purpose of RC is to measure the reusability of components in order to realize the reuse of component effectively and to identify the best components in terms of their reusability [24].

4. Characteristics of Component Evaluation Approaches

Based on exhausted literature, following are brief explanation of characteristics for component evaluation approaches that are combined from PLC, OC, QC and RC approaches. As a result, these are 15 characteristics for components evaluation approaches that are adopted from these approaches such as functionality, component replaceability, functional commonality, applicability, non functional commonality, variability richness, tailorability, understandability, reliability, usability, efficiency, maintainability, portability, adaptability (flexibility), and confidence/probability [7, 17, 18, 24-27].

4.1 Functionality

Functionality expresses the ability of a component to provide the required services, when used under specified conditions. Under functionality there are many sub-characteristics such as interoperability, completeness, accuracy, suitability, correctness, compliance, and security.

4.2 Component Replaceability

This characteristic can be used in measuring the capability to replace embedding components by in-house or COTS components without changing the architecture. The sub-characteristic for component replaceability is embedding components.

4.3 Functional Commonality

This characteristic is to measures the functions provided by core asset if they are common to the applications defined in the PL scope. Sub-characteristic for this characteristic is common functionality.
4.4 Applicability
This characteristic uses to measures the capability of core asset to be applied by a number of family members. Similar to functionality, applicability has two sub-characteristics known as variability and common functionality.

4.5 NonFunctional Commonability
This characteristic measure the non-functional requirement provided by core asset if it is common to the members defined in the PL scope. The sub-characteristics for this characteristic are providing architecture genericity and application level core asset.

4.6 Variability Richness
This characteristic is used to measures the core asset if it is widely comprehend the variability in a PL scope, that means if the boundary of variability is confined, then it will be reused by a restricted number of applications. These are two sub-characteristics for this characteristic namely variability and open variability

4.7 Tailorability
Tailorability is to measures the capability of the core asset to be feasibly tailored adhering to the application specific requirements or to tailor a core asset, PLE uses instantiation mechanisms. The only one sub-characteristic for tailorability is instantiation.

4.8 Understandability
This characteristic expresses the understanding for functionality of the component in order to decide whether it meets the new functional requirement. In PL concept, the understandability is used to measures the capability of a core asset to be easily understood and learned when developing applications based on core asset. Complexity, documentation level, existence of meta-information, observability, and instantiation are the sub-characteristics for the understandability.

4.9 Reliability
Reliability shows the ability of the component to maintain a specified level of performance, when used under specified conditions. Similar to other characteristics reliability has their own sub-characteristic e.g. maturity, compliance, service stability, recoverability, result set, and fault tolerance.

4.10 Usability
Usability expresses the ability of a component to be understood, learned, used, configured, and executed, when used under specified conditions. The sub-characteristics for usability are installability, monitorability, compliance, configurability, learnability, help tool, operability, and identifiability-reachability.

4.11 Efficiency
Efficiency explains the ability of a component to provide appropriate performance, relative to the amount of resources used. There are several sub-characteristics under efficiency namely compliance, time behaviour, resource behaviour, scalability, response time, and system overhead.

4.12 Maintainability
Maintainability is defined the ability of a component to be modified. It has nine sub-characteristics such as analyzability, stability, changeability, testability, customizability, diagnoseability, repairability, and compliance.

4.13 Portability
Portability describes the ability of a component to be transferred from one environment to another or porting the component to a new environment. Similar to maintainability, portability also has own sub-characteristics namely adaptability, replaceability, compliance, deployability, and external dependencies.

4.14 Adaptability (Flexibility)
Adaptability is used to explains the adapting of the component to the specific functional that required to the new system. These are sub-characteristics for adaptability e.g. generality, modularity and customizability.

4.15 Confidence/ Probability
Confidence is associated to a single criterion, framework design maturity or simply maturity that comes from the number of faults detected during framework design reviews and from the requests for changes or new features addressed. Maturity is the only one sub-characteristic for confidence.

5. Comparison of Component Evaluation Approaches
This section describes a comparison of the component evaluation approaches introduced in the previous section. The evaluation primarily focuses on three criteria i.e. a scope that concentrates on the types of the considered components, technique that refers to the way metrics are defined, and level of validation which covers the experiments that have been done for the validation of the approaches. The initial results obtained from this evaluation can be used to indicate to which extent a particular approach satisfies the necessary characteristics in
term if its support for software component reuse. The summary of the comparison results are shown in Table 2. In Table 2, the ‘tick’ indicates that the approach address the related sub-characteristic and the number (inside the bracket) indicate the number of weight for approaches.

5.1 Scope

As mentioned earlier, scope refers to the types of components that the approaches covered. For PLC it only covered the components of product line [16, 17]. OC approach only covers original components which have a narrow scope with the primary task to provide functionality that performs specific actions [7].

QC approach mainly covers COTS components [20, 28, 29], and according to Gill and Grover, this approach also covered black box component [30]. Research has shown that RC approach is used to evaluate white box Java Beans [31] and in practice it also used in black box Java Beans Interface [24].

5.2 Technique

Technique is referred to how the metrics are defined and applied [32]. Based on this evaluation criterion, the approach can be classified into the following categories: Wish list – The authors informally identify the need for a certain kind of metrics, without actually proposing any metrics, Informal – A natural language description of the metrics is provided by the authors, Semi-formal – Some degree of formalism is used in the metrics definitions. Typically, the metrics themselves are defined through mathematical expressions, but the underlying concepts being measured are only informally specified, and Formal – A formal definition of the metrics and underlying concepts is provided.

In PLC approach, Hoek uses semi formal technique in order to define the metrics [33] whilst Her et al. use formal technique in their metrics definition [17]. OC approach applies informal technique in order to define its metrics [7].

Based on the literatures, QC approach use informal technique [20, 29] and wish list [28, 30] to define the metrics. RC approach applies semi formal technique [24] and informal technique [31].

Based on the technique used for these approaches, it can be concluded that PLC approach has a concrete metric definition compared to other approaches.

5.3 Level of Validation

In practice, Goula¬o proposed a set of rules to be used in comparing the level of validation (LV) of the approaches [32]. It can be classified according to the following categories: Anecdotal – Anecdotal examples are provided to motivate the usefulness of the proposed metric. Sometimes, they are complemented with some descriptive statistics. Small experiment – An experiment is carried out to assess the metrics, with some statistical approach to analyze the collected data, but the sample of analyzed artefacts does not allow generalizing conclusions. Industrial experiment – An experiment with a significant sample of artefacts is carried out, with real-world artefacts and adequate statistical analysis. Independently validated – Experiments made by independent research teams confirm the conclusions and claims made by the metrics proponents.

The level of validation for the reviewed PLC approaches is low, for examples anecdotal [33] and small experiment [17]. OC approach can be considered as a middle level of validation. It demonstrates the applicability of their approach using industrial strength real software application [7]. Most of the QC approaches do not provide a clear evidence of validation [20, 28, 29] except the QC approach proposed by Alvaro et al. that have accomplished a preliminary evaluation for the component marketplace [27]. RC approach shows evidences of two level of validation i.e. anecdotal [31] industrial experiment [24].

From the result that mentioned above, all approaches that it reviews still not achieve the high (independently validated) level of validation.
6. The Comparison Result

From the analysis on the component evaluation approaches, every approach has their own characteristics. These characteristics are used in their process of component evaluation for the approach. Table 2 summarizes the evaluation results based on the proposed evaluation criteria. These results show that PLC has a higher maturity, followed by RC, OC and OC approach. Even thought PLC approach propose the evaluation for Product Line Engineering but it still has limitation because it only applied the approach on a few projects. There is a lack of validity test over a sufficient number of other core asset product line engineering applications.

RC approach also used a metric suite for measuring reusability of software components which are suite for black-box component based on limited information that can be obtained from the outside of the component without any source code.

From the comparison evaluation results, all approaches that it reviews in this study still not achieve the high (independently validated) level of validation.

Based on the discussion mention above, the conceptual framework for this study will be proposed in order to get a new component evaluation approach.

7. Conclusion and Further Work

In this paper, we have presented the state-of-the-art component evaluation approaches, especially in the context of software reuse. We have evaluated the approaches using the appropriate criteria for the comparative evaluation of reusable components. The results showed us that so far, there is no approaches fully satisfied all of the requirements of component evaluation. This means that much work have to be done to achieve the better approaches in the future.

Currently, we are developing a conceptual framework of our component evaluation approach that will be applied to a selected industrial strength case study. An automated tool that supports the proposed conceptual framework for the approach will be developed to demonstrate the possible automation of the proposed approach. Both the proposed approach and the automated tool will be empirically validated. It is expected that the results of this research may help software engineers to select an appropriate reusable components for their software project, and determine the quality of the developed reusable components.
References
