A Survey of Methods for Measuring and Enhancing Component Reusability

Ditmar Parmeza  
2-Year Master in Software Engineering  
Mälardalen University  
Västerås, Sweden  
dpa12001@student.mdh.se

Miraldi Fifo  
2-Year Master in Software Engineering  
Mälardalen University  
Västerås, Sweden  
mfo12002@student.mdh.se

ABSTRACT
One of the most interesting issues in Component-Based Software Engineering is definitively component reusability. Component reuse is an approach that provides many advantages towards the process of building new components for large complex systems. Component reusability consists of some functionalities related to components such as adaptability, reliability and other features. These features are very important for the software component reusability evaluation.

Some research papers have proposed several solutions for evaluating component reusability in the best possible way. Most of them have introduced methods that provide an accurate estimation of reusability and ways how to enhance it further. In this paper, we review three of these methods: COR (Component Overall Reusability), Coupling-Cohesion and CRECOR (Component Repository for Facilitating Enterprise JavaBeans Component Reuse). Our goal is to analyze the advantages and disadvantages of each method. Moreover, we focus on comparing qualitative aspects of reusability such as adaptability or reliability as well.

It is expected that CRECOR should provide more advantages in terms of reusability features due to the fact that it includes some extra-functionals that are missing in COR and Coupling-Cohesion methods. However, each method has its own strong points that will be analyzed in the paper.

General Terms

Keywords
Component, Reusability, Methods, Metrics, COR, Coupling, Cohesion, CRECOR, Repository, Adaptability, Qualitative.

1. INTRODUCTION
Reusability is a very important concept in Component-Based Development which implies the extent to which a reused component can be suited for working correctly in a brand new system. A component with a high level of reusability can decrease the cost of the software system that is going to use it. It can save time as well since it eliminates the need for developing a new component which would require more work effort and also time [2][14][15]. In order to reuse these components in a new software, we need to measure their reusability level [5]. In this case, there are some problems since many methods used for this estimation usually require the source code of the main components. The existence of black-box components increases the need to perform measurements without analyzing their source code [3]. There is also research work which focuses on software component models [4] that facilitate the reuse of components but we will focus mostly on methods that aim to make a proper estimation of component reusability and contribute into its enhancement.

A component must be designed appropriately when it is previously known that it will be reused [17]. In this case, an appropriate way would be to design the component according to an approach as generalized as possible so it can be suited for future use as well. A well-known implemented component model that relies on reusability is the Koala model [1]. Meanwhile, CORBA (Common Object Request Broker Architecture), EJB (Enterprise JavaBeans) and other component technologies have already been successful into embracing component reusability [16].

This paper aims to make a survey of three methods that estimate component reusability and contribute on improving it. By making this survey, we want to underline the pros and cons of each method. CRECOR is the method that provides more facilities for the process of reusing components. Therefore, it is expected that this method results to be more successful. However, advantages of COR and Coupling-Cohesion methods towards CRECOR do exist.

The rest of the paper continues with section 2 which describes some related work. In section 3, we explain the method or the approach that we used and followed in order to prepare this paper. The content of section 4 includes an overview of each method. Section 5 contains a review of performance evaluation of each case. A summary of the main ideas is provided in section 6. Section 7 is about the main conclusions of the work while some information related to future work can be found in section 8.
2. RELATED WORK
Several methods for providing ways how to measure and improve reusability have been proposed in papers [6-13]. For example, Washizaki et al. introduce COR in [6] as a combined approach of five previous metrics to guarantee a higher level of reusability. A comparison is performed between the previous five metrics and all of their advantages are combined together in COR in order to assure a better performance. Meanwhile, Gui and Scott present Coupling and Cohesion method for reusability evaluation in [7] and improve it further in [8]. These papers provide enough convincing arguments to show the important role and advantages of Coupling-Cohesion towards conventional metrics in terms of evaluating component reusability.

Papers [9-13] focus on CRECOR and its advantages. CRECOR is presented as an approach that totally relies on a repository technology. Lee et al. explain in [9] the way how CRECOR makes the component re-use process easier. Experimental results are presented in [13].

All these papers have in common the fact that they try to show the benefits of their methods regarding reusability evaluation. They compare the proposed method(s) with established ones and analyze their benefits. The biggest difference between these papers and our paper is that we analyze the following methods altogether: COR, Coupling-Cohesion and CRECOR. Our goal is to review them in order to analyze their strong and weak points related to reusability aspects.

3. METHOD
We used specific search terms in order start searching for literature that would help us on writing the paper. Google Scholar was the main used search engine. The most important keywords that were used were “Reusable Components”, “Reusability Issues”, “Component Reusability Evaluation”, “Reusability Metrics”, “EJB Reusability”, “Improve Component Reusability” and “Component Reusability Methods”.

After a considerable number of references (30) were collected, we proceeded with a selective process in order to choose the papers that seemed to be more useful for the topic. At the end, 17 references were considered useful for making this survey. The criteria for selection was the existence of accurate information related either to one of the methods that we analyze in this paper (COR, Coupling-Cohesion, CRECOR) or to some background regarding reusability issues.

We chose to include these methods in our paper since there are relevant research studies that present them and analyze the way how they are superior towards conventional methods. However, we aim to analyze them in respect to each other. For example, we try to conclude on defining which method has the best contribution in component adaptation, reliability and other qualitative features related to component reusability.

4. OVERVIEW OF METHODS FOR REUSABILITY EVALUATION
An overview of each method is presented in the next subsections.

4.1 Overview of COR
COR (Component Overall Reusability) is a combination of some metrics that evaluate component reusability. The method is introduced by Washizaki et al. in [6]. Its purpose is to measure black-box component reusability. Such components do not provide too much information (e.g. source code). The information needed is ensured from the external part of components e.g. interfaces. The method is intended to work on JavaBeans architecture.

The metrics measure only the features that affect component reusability. Cohesion and coupling are part of that group of features but they are not used by COR metrics since they cannot be analyzed by metrics that do not access the source code. There are three features that affect reusability: understandability, adaptability and portability [6]. Understandability refers to the degree of component complexity which affects the way how the user understands the concept of the main component. Adaptability estimates the degree of component adaptation to a different environment that differs from the one where the component was developed for the first time. Portability estimates the easiness or the complexity of the situation when the component needs to be moved from one architectural environment to another one.

There are five metrics included in COR [6] that affect all the characteristics mentioned above. Each metric has an interval of confidence. If the value of the metric is close to the upper limit of the interval, then we can say that the characteristic that we are measuring has a high quality. The first metric is called Existence of Meta-Information (EMI). The metric shows the existence or absence of a class called BeanInfo that provides meta-data information. Its value can be 1 or 0. Its confidence interval is [0.5, 1.0]. The confidence interval includes the reusability predicted levels which are estimated with high precision by the metric. For example, when EMI predicts a reusability level of 0.7 (70%) which is included in its confidence interval, the predicted result is expected to be highly precise i.e., close to the real value. On the other hand, if EMI predicts a reusability level of 0.2 (20%), its reliability is quite low since this value is not included in the confidence interval. Rate of Component Observability (RCO) is the second metric. It shows how many readable properties exist in the Facade class which is a class that includes read methods. In other words, it measures the component observability level. Its confidence interval is [0.17, 0.42]. The third metric is Rate of Component Customizability (RCC). It shows how many writable properties exist in the Facade Class. Therefore, it estimates the component customizability level. Its confidence interval is [0.17, 0.34]. The fourth metric is Self-Completeness of Component’s Return Value (SCCr). This metric shows how many methods do not return any value compared to all the implemented methods in a component. This metric evaluates the external dependency of the component. If the number of methods that do not return a value is small, then the level of component external dependency is significantly low. The confidence interval of this metric is [0.61, 1.0]. The fifth metric is Self-Completeness of Component’s Parameter (SCSp). It shows how many methods do not have any parameter compared to all the implemented methods in a component. As the previous metrics, it evaluates the external dependency of the component. Its confidence interval is [0.42, 0.77].

COR summarizes all the metrics above and it keeps track of the component reusability level. Three different metrics such as EMI, RCC and SCCr are combined in order to evaluate respectively component understandability, adaptability and portability [6].

\[
COR(c) = 1.76 \times \frac{V_{EMI}(c) + V_{RCC}(c) + V_{SCCr}(c)}{3} - 1.13
\]  

If the value of COR(c) is larger than 0, we can say that the component reusability level is high. These metrics have two
advantages towards previous established metrics. They can be used for black-box components and for components that have been recently developed.

4.2 Overview of Coupling-Cohesion Method

Coupling-Cohesion method represents another method that is used for evaluating software component reusability. This method is introduced in [7-8]. It has been developed by the authors in order to make an accurate reusability estimation of Java components that are retrieved from the Internet by a search engine. It was expected to obtain an enhancement of performance towards pre-established methods by the time that Coupling-Cohesion method was introduced ([7] in 2006 and [8] in 2008). The main purpose of this method is to improve software component reusability by focusing on two important component aspects: reliability and adaptability. Reliability must be in a good level so the component can be used in more complex software systems. Otherwise, including a not very reliable component into a big system can usually have unacceptable consequences e.g., the whole system may be damaged or not work at all. It is also crucial to have a satisfying level of adaptability so the component can be easily adapted for being used in a new system [7].

Coupling and Cohesion are two concepts that help on defining and evaluating component reusability in two different ways. However, both of them are related to component design issues. Coupling is an approach that keeps track of the level of interaction between the subcomponents within a component. The level of interaction should be as low as possible. Otherwise, changing one subcomponent would have a negative effect on all the subcomponents that have a good interaction with it. Consequently, having coupling in a high level results in a low reusability level. On the other hand, cohesion is an approach that estimates the similarity between the functions that are performed by a subcomponent. In order to have an optimized distribution of functionalities i.e., resulting in a higher reusability level, the cohesion must be as high as possible. It is clear that a linear dependency exists between cohesion and reusability i.e., increasing one of them brings to the increase of the other one [7][8].

Furthermore, Gui and Scott analyze these concepts in [8] in the context of methods and classes. To be more precise, coupling estimates the level of interaction between classes while cohesion estimates the similarity between the instance variables that are accessed by the methods that are part of a class. The whole system is seen as a directed graph where vertices represent the classes. If one class would invoke many methods that are also invoked from another class, then coupling would be significantly high and it would have a negative impact on reusability. Coupling is estimated by using the following relation [7][8]:

\[
WTCoup = \frac{\sum_{i,j=1}^{m} Coup(i,j)}{m^2 - m}
\]  

Relation (2) implies that WTCoup (Weighted Transitive Coupling) is estimated by diving the total sum of couplings of all pairs of classes in the system by \(m^2 - m\) where \(m\) is the number of classes in the system. In a similar way, a relation for WTCoh (Weighted Transitive Cohesion) has been obtained and showed in [7] and [8]:

\[
WTCoh = \frac{\sum_{j=1}^{n} ClassCoh(j)}{n}
\]  

Relation (3) implies that WTCoh is estimated by diving the total sum of couplings of classes in the system by \(n\) where \(n\) is the number of classes in the system.

An experimental comparative evaluation of Coupling and Cohesion method proposed by Gui and Scott and previous established methods is performed in [8] in order to show the success of the new method in terms of performance. Three different cases (HTML Parser, Lexical Tokenizer and Barcode Generator) are analyzed for the components and two different approaches (Linear regression and Spearman Rank Correlation) are used for making the quantitative evaluation of the methods. In all cases, the highest reusability estimation rate (80-90%) is achieved by the new Coupling-Cohesion method which significantly overpasses the performance of previous methods. More details are provided in subsection 5.2.

4.3 Overview of CRECOR

In this subsection, we provide an overview of CRECOR [9-13] which is the third method that will be part of this survey. This method consists in using a component repository in order to make the process of reusing components less complex. Advantages of this repository are explained in [9]. CRECOR provides many integrated functionalities that are missing in other approaches. These functionalities involve different phases of component reusability such as specification viewing, adapting, testing and deploying [9][12].

The main idea of the component repository is that it includes several types of components that have been previously used in different systems. When there is the need for reusing any of them for a new system, the repository provides the opportunity to search for certain components and browse all of them that are in the repository. After choosing the component(s) that is (are) going to be reused, the repository provides the functionality for adaptation i.e., making the new component more suitable for being used in the new system. This also facilitates the process of composition and component integration in the system. This is possible thanks to the facilities that the repository provides in terms of searching and finding the most suitable component in the repository.

The technology used in the research work presented in [9] is EJB and the main goal of this study is to make EJB components easier for reuse. Indeed, introducing CRECOR contributed to a considerable reusability enhancement of EJB components. Beside the functionalities mentioned above, CRECOR provides non-functional features as well e.g., components can be dragged and dropped in the list of reusable components. One of the most notable advantages of CRECOR is its own GUI (graphical user interface) which provides support for the component reuse process.

Despite the fact that the introduction of CRECOR in 2003 was a big contribution on component reusability enhancement, it needed some further improvements. CRECOR is implemented for an IP Multimedia Subsystem-Based (IMS-Based) mobile application that is presented in [10]. In order to ensure a higher level of reusability, some functionality such as adaptability has been improved. The component repository introduced for CRECOR in [9] was considered as a very good approach for supporting the
Further improvements on CRECOR method are proposed in [11]. This paper presents a new version of the component repository which basically relies on the first version that was previously introduced in [9] and it also adds the possibility for controlling and handling different versions of the reusable components. This new version is used for a new Component-Based Software Engineering approach that is developed for handling software system development by either building components from the beginning or using previously used ones. Another aspect that has been improved in [11] is related to the issue of selecting the components that should be reused. In previous methods and approaches, this selective process was done usually in the design phase. Making the selection of reusable components during this development phase did not bring to satisfying results i.e., a high component reusability level since a deeper analysis of the components was missing before continuing with the selection process. Fortunately, the new CBD (Component-Based Development) model includes this analysis phase and the selection of reusable components is performed precisely during this phase.

Other functional aspects that contribute in component reusability enhancement include classification i.e., the used components are categorized in the repository in order to facilitate the process of browsing and selecting components for reuse. Modification and adaptation are other functional aspects that are improved thanks to the component repository.

A separate development phase dedicated only to adaptation issues is part of the CBD model in [11]. After the components have been selected to be reused, they must be modified and improved in order to fulfill the functional or non-functional requirements of the new system. Once this objective has been accomplished, it is time to deal with the process of component composition and integration [1][9][11]. This is one of the most crucial and problematic cases because it handles problems with interactions between several components and their integration in the new system that is being developed.

Finally, testing is another important functionality supported by CRECOR [9]. Testing is always performed in the end in order to make sure that the reused components work correctly in the new context.

Different alternatives of how the components can be browsed and selected from the repository are analyzed in [12]. According to this analysis, the best alternative is the semantic-oriented method which provides the possibility to browse components in the repository and select one or some of the components for reuse by entering the component name and prototype. This alternative brings to a fast search i.e., it is time-effective and reduces the complexity of the whole reusability process.

We already mentioned that GUI is one of the big advantages of CRECOR. The implemented version of the method provides a very user-friendly interface that visualizes and comprises all the necessary functionalities [9]. For example, the adaptation problem is resolved by an interface that provides options for adding, modifying, replacing or deleting methods and attributes related to existing components.

5. Performance Review of Each Method
The next three subsections present a review of studies made for evaluating the performance of each method. Since different approaches have been used in each case, it is hard to quantifiably compare the three methods together in terms of performance. However, we can review each method separately and check which approach has brought to the biggest success.

5.1 COR Performance
In order to review performance evaluation of COR method, we refer to the experimental results presented in [6]. As it was already stated in subsection 4.1, formula (1) is used in order to estimate the reusability level of software components.

The value of COR(c) needs to be positive in order to assure that certain software components can be reusable. This implies that component understandability, portability and adaptability should be in relevant levels. An experimental study of black-box component reusability takes into account a component that consists of 13 read methods and 13 write methods. Three versions of the component are analyzed in terms of reusability. The first version (V1) includes all the methods mentioned above meanwhile the second one (V2) does not include some methods that are not necessary for some component functionalities. The third version (V3) eliminates all the unnecessary methods. The results for each version are shown in figure 1 [6].

![Figure 1: Performance values for COR (3 versions)](image)

The performance is significantly improved in versions V2 and V3. The reason is that V2 and V3 include only the methods that are directly related to important component functionalities. Having fewer methods than V1 gives them the advantage to adapt easier in a new system. Consequently, component reusability is higher. Thus, the best result of COR performance consists in a level of 63% of component reusability.

5.2 Coupling-Cohesion Performance
Performance of Coupling-Cohesion method is evaluated by following different approaches [7-8] from the one used for COR. These approaches are Linear Regression and Spearman Rank Correlation. Linear Regression estimates reusability by measuring the exact number of lines of code that need to be modified, added or removed when a component should be prepared for being reused in a new system. The second approach, Spearman Rank Correlation ranks components that are retrieved from the repository according to their reusability level. The Coupling-Cohesion method presents a higher performance than conventional methods. The experimental studies in [7-8] include three cases for evaluating this method. Three different objects of
analysis are taken into consideration in each case. The first one uses an HTML parser, the second one relies on a lexical tokenizer and the third one analyzes a barcode generator. The results for each case are summarized in figure 2.

![Figure 2: Performance values for Coupling-Cohesion method](image)

The results obtained for Coupling (shown in blue) and Cohesion (shown in red) by using both Linear Regression (L.R.) and Spearman Rank Correlation (S.R.C.) are shown in figure 2. WTCoup (Weighted Transitive Coupling) and WTcoh (Weighted Transitive Cohesion) are used to represent respectively the evaluated component coupling and cohesion. As we already stated, a low coupling contributes into a high reusability level. Meanwhile, a high cohesion is desirable. The highest cohesion in figure 2 is estimated to be 89.2% while the lowest coupling is 76.1%.

5.3 CRECOR Performance

CRECOR is the third method that will be evaluated in terms of performance. In paper [13], four cases are studied in order to check how components are searched and retrieved from the component repository. These cases include text-based, keyword-based, signature-based and semantic-based search method. The highest performance is observed in the last method i.e., semantic-based search method. This method provides the possibility to search for a component by entering its name and its prototype. It facilitates the process of searching and retrieving the component from the repository. Of course, the component is ready for reuse only after it has been successfully retrieved.

Precision and recall are the two units used in [13] for defining the level of a successful component retrieving. When both of them are 1 (100%), that means that each part of the repository has been checked in order to see if there is any match with the data provided for the searched component. As we already mentioned, the semantic-based method provides the highest performance. Experimental results regarding the precision and recall level of CORBA, EJB and COM+ components are shown in figure 3.

![Figure 3: Precision and Recall Rate for CRECOR](image)

The results are more than positive. Precision is 1 (100%) in all cases. Recall is at a 50% level when the technology used is either CORBA or COM+. However, both precision and recall are at their maximal values (100%) when EJB components are tested. In this paper, we are more interested in the results related to EJB components since the research studies made for COR and Coupling-Cohesion methods also rely on EJB technology.

Although we cannot compare directly the results produced for each method, we can see that the most successful performance results are assigned to CRECOR. If EJB components are used, it has been proved that they can be searched and retrieved from the component repository with the highest possible precision. All this component search and retrieval process is done in order to reuse the selected components. Therefore, a high performance on searching and retrieving directly contributes to component reusability enhancement.

6. SUMMARY

The main goal of this paper was to analyze three methods that contribute on estimating and enhancing component reusability. The motivation for making this survey is shown in the introduction section of the paper by underlining the importance and benefits of reusing components. Reusing components eliminates the cost that would be needed for building a new component and decreases the development process time.

We provided an overview of COR, Coupling-Cohesion and CRECOR in section 4.

CRECOR represents generally the most successful method out of the three ones that are overviewed in this paper. It relies on a component repository which supports many functionalities such as specification viewing, component adaptation, analysis, design, integration, testing and GUI.

A performance review of each method is presented in section 5. It is impossible to confront the experimental results directly since different approaches and metrics have been used in each case. However, the best performance rate has been reached by
CRECOR since the precision level is 100% regardless if the technology used is EJB or another one. In fact, CRECOR is superior towards the other two methods in many aspects.

7. CONCLUSIONS

This paper did neither introduce nor present a new concept, method or approach but it consists in a review of three established methods that evaluate component reusability and focus on its further enhancement.

As expected, CRECOR represents the most convenient method. Much effort has been put into improving this method in the last decade. It is hard to state that CRECOR has the best performance since the results presented in section 5 are not comparable. However, it manifests the biggest success. This result is related to the fact that most of the reusability parameters in CRECOR are significantly higher than the ones in COR and Coupling-Cohesion methods. These parameters include adaptability, understandability and portability. The process of component adaptation is much easier in CRECOR because of the functional and non-functional supports provided by its repository. The one that provides the biggest contribution to component adaptability enhancement is the analysis phase which gives the possibility to select components before design is started. Understandability and portability are also higher in CRECOR. It is easier to understand the main concept of a component since CRECOR repository gives the possibility to have a list of reused components according to their names and prototypes. A higher level of portability is present in CRECOR because the GUI provided by this method facilitates the process of moving components from one architecture to another one.

Meanwhile, the Coupling-Cohesion method is much superior in terms of component reliability due to the detailed analysis of the interactions between subcomponents that is made by the coupling approach. Having more detailed information on the internal structure of the component increases the reliability and the confidence to reuse that component in a new software system.

On the other hand, COR has its own advantage as well which is related to the fact that this method can be used for black-box components. As a result, there is no need for component source code analysis.

8. FUTURE WORK

One possible alternative for future work would be to propose a new method that would consist in combining the three methods altogether: COR, Coupling-Cohesion and CRECOR. The best option would be to include only the advantageous aspects of each method and to avoid the disadvantageous ones. This way, the method would ensure that component adaptability, understandability, portability and reliability are all in quite high levels. Moreover, reusability would be possible for different kinds of components including white-box, gray-box and black-box ones. Another interesting alternative for future work would be to provide our own metrics for making a quantifiable comparison of the three methods. Consequently, we would be able to compare directly their performance. However, it is expected that CRECOR would provide the best performance even in this case due to the advantages analyzed in this paper.

9. ACKNOWLEDGMENTS

This paper is written as part of an assignment for the course “Research Methodologies in Natural Sciences and Engineering” at Mälardalen University in Sweden. Ideas and advices on how to write the paper have been provided during the lectures by the professors Gordana Dodig-Crnkovic and Jan Gustafsson. Some information regarding Component-Based development given by prof. Ivica Crnkovic in the “Advanced Component-Based Software Engineering” course at the same university has also been used.

10. REFERENCES


