Software Component Certification: A Component Quality Model

Alexandre Alvaro, Silvio Lemos Meira

Federal University of Pernambuco and C.E.S.A.R – Recife Center for Advanced Studies and Systems, Brazil

{alexandre.alvaro, silvio.meira}@cesar.org.br

Abstract. *Component-based software development is becoming more generalized, representing a considerable market for the software industry. However, several technical issues remain unsolved before the software components industry reaches the maturity as other software industries. Problems such as component selection and the uncertain quality of third-party developed components bring new challenges to the software engineering community. In contrast, software component certification is still immature and much research is needed in order to create well-defined standards for certification. This paper introduces a component quality model, based upon consistent and well-defined quality characteristics, and describes a formal case study that was used in order to analyze the viability of the model usage.*

Resumo. *Desenvolvimento de Software Baseado em Componentes tem sido amplamente adotado, representando assim um mercado promissor para a indústria de software. Entretanto, inúmeros problemas técnicos ainda permanecem sem solução antes mesmo que a indústria de componentes de software alcance a maturidade de outras indústrias de software. Problemas como a seleção de componentes e a falta de informações sobre a qualidade dos componentes desenvolvidos trazem novos desafios para a comunidade de engenharia de software. Por outro lado, a área de certificação de componentes de software é relativamente imatura e necessita de consideráveis pesquisas para o estabelecimento de um padrão para certificação de componentes de software. Assim, este artigo apresenta um modelo de qualidade de componentes, baseada em características consistentes e bem definidas e, apresenta um estudo de caso formal o qual visa analisar a viabilidade de utilização do modelo.*

1. Introduction

One of the most compelling reasons for adopting component-based approaches in software development, with or without objects, is the premise of reuse. The idea is to build software from existing components primarily by assembling and replacing interoperable parts. The implications for reduced development time and improved product quality make this approach very attractive [Krueger, 1992].

Reuse is a “generic” denomination, encompassing a variety of techniques aimed at getting the most from design and implementation work. The top objective is to avoid reinvention, redesign and reimplementation when building a new product, by

capitalizing on previous done work that can be immediately deployed in new contexts. Therefore, better products can be delivered in shorter times, maintenance costs are reduced because an improvement to one piece of design work will enhance all the projects in which it is used, and quality should improve because reused components have been well tested [D'Souza et al., 1999], [Jacobson et al., 1997].

Software reuse is not a new idea. Since McIlroy's pioneer work, "*Mass Produced Software Components*" [McIlroy, 1968], the idea of reusing software components in large scale is being pursued by developers and research groups. This effort is reflected in the literature, which is very rich in this particular area of software engineering.

Most of the works follow McIlroy's idea: "*the software industry is weakly founded and one aspect of this weakness is the absence of a software component sub-industry*" (pp. 80). The existence of a market, in which developers could obtain components and assemble them into applications, was always envisioned.

On the other hand, these works do not consider an essential requirement for these systems: the assets certification. In a real environment, a developer that retrieves a faulty component from the repository would certainly lose his trust on the system, becoming discouraged to make new queries. Thus, it is extremely important to assert the quality of the assets that are stored into the repository before making them available for reuse. Despite this importance, the software engineering community had not explored these issues until recently. In this way, a new research area arose: components certification and quality assurance [Wohlin et al., 1994], [Morris et al., 2001], [Wallnau, 2003]. However, several questions still remain unanswered, such as: **(i)** how certification should be carried out? **(ii)** what are the requirements for a certification process? and, **(iii)** who should perform it? [Goulão et al., 2002]. This is the reason why there is still no well-defined standard to perform component certification [Morris et al., 2001].

In this context, the main goal of this work is investigating effective ways to demonstrate that component certification is not only possible and practically viable, but also directly applicable in the software industry. Through certification, some benefits can be achieved, such as: higher quality levels, reduced maintenance time, investment return, reduced time-to-market, among others. According to Weber et al. [Weber et al., 2002], the need for quality assurance in software development has exponentially increased in the past few years. This fact could be seen through a nationwide project launched by the Brazilian government[1]. This project's main concerns are: to develop a robust framework for software reuse [Almeida et al., 2004], in order to establish a standard to the component development; to develop a repository system; and to develop a component certification process. This project has been developed in a collaboration between the industry and academia (the RiSE group[2] and two other universities), in order to generate a well-defined model for developing, evaluating software component

[1] http://www.finep.gov.br//fundos_setoriais/acao_transversal/resultados/resultado_Acao_transversal_Biblioteca_de_Componentes_05_2004.PDF *(in portuguese)*

[2] RiSE (Reuse in Software Engineering) group – http://www.rise.com.br

quality, storing and, after that, making it possible for software factories to reuse software components.

1.1. Software Components Inhibitors

To assess the market for Component-Based Software Engineering (CBSE), the Carnegie Mellon University's Software Engineering Institute (CMU/SEI) studied industry trends in the use of software components. The study [Bass et al., 2000], conducted from September 1999 to February 2000, examined software components from both technical and business perspectives.

A distinct set of inhibitors to adopting software component technology emerged from the conducted surveys and interviews of earlier adopters of software component technology. A summary from a Web survey of component adopters is presented in Figure 1.

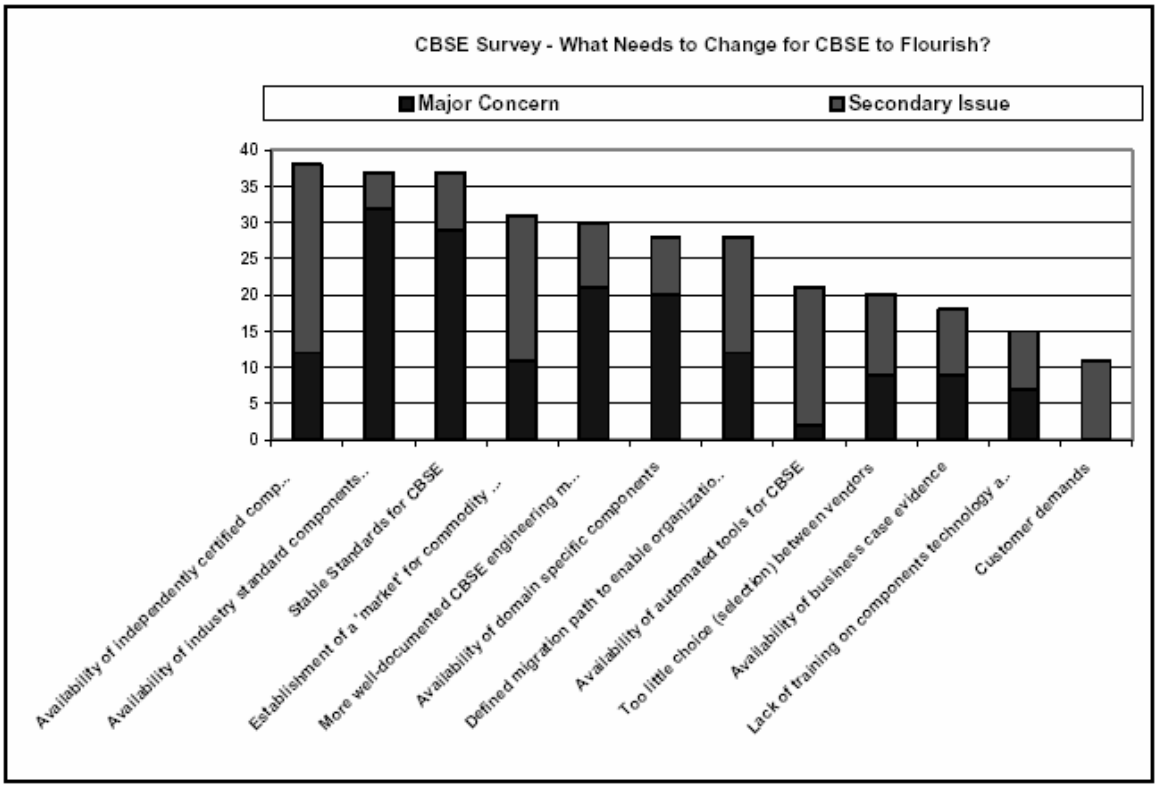

Figure 1. Summary of Survey Responses [Bass et al., 2000].

From this data and from the interviews, the study concludes that the market perceives the following key inhibitors for component adoption, presented here in decreasing order of importance:

- Lack of available components;
- Lack of stable standards for component technology;
- **Lack of certified components**; and
- Lack of an engineering method to consistently produce quality systems from components.

The software engineering community is already attempting to reduce the gaps related to the two first inhibitors. A look at the main Internet component markets, such

as *Flashline*[3] and *ComponentSource*[4], which contain, respectively, more than 13,000 and 9,000 components (four years ago they had less than 1,000 and 100 components [Trass et al., 2000], respectively), leads to conclude that there is a large increase in the availability of reusable assets. In the same period, component technologies have obtained considerable maturity, especially those related to JavaBeans, Enterprise JavaBeans (EJB), and Microsoft .NET technologies. Thus, the software engineering community is trying to establish stable standards for component technology, each one for a particular market niche.

However, in relation to the third inhibitor, the community is still a fledgling. Further research is required in order to assure the production of certified components, especially when combined with the lack of component-based software engineering techniques that deliver predictable properties (the last inhibitor).

The concern with components certification reflects a natural progression of concerns: first demonstrate that it is possible to build and sustain a component-based system at all, and then improve the overall quality of components and the consumers' trust in these components.

Still on, according to Voas [Voas, 1998], to foster an emerging software component marketplace, it must be clear for buyers whether a component's impact is positive or negative. Ideally, buyers should have this information before buying a component. Component buyers could then choose an appropriate component according to its certification level. With this information, system builders could make better design decisions and be less fearful of liability concerns, and component vendors could expect a growing marketplace for their products.

1.2. The Future of Software Components

Important researches on software components, such as Heineman [Heineman et al., 2000], Councill in [Heineman et al., 2001], Crnkovic [Crnkovic, 2001] and Wallnau [Wallnau, 2003] points to the future of software components is certification. These authors state that certification is a necessary precondition for CBSE to be successfully adopted and to achieve the associated economic and social benefits that CBSE could yield. With the success of CBSE, software developers will have more time to develop, instead of spending their time addressing problems associated with understanding and fixing someone else's code. Certified components used during development will have predetermined and well-established criteria, thus reducing the risk of system failure and increasing the likelihood that the system will comply with design standards.

When the system is developed using a CBSE approach, the use of certified components could provide objective evidence that the components meet rigorous specifications including data on intended use. This approach does not permit the designer to forego inherently safe system design practices. Instead, certification reduces the risk of system failure by providing information about a software component's risk mitigation procedures, such as the anticipation about the software failure state and return to the last stable state with notice to the system administrator. The objective is to

[3] http://www.flashline.com
[4] http://www.componentsource.com

build safe systems from well-documented and proven components. And if these components are independently certified, the confidence that the information accompanying these components meets their requirements will be greater.

2. Software Component Certification: A Brief Survey

Existing literature is not that rich in reports related to practical software component certification experience, but some relevant research explores the theory of component certification in academic scenarios. The timeline can be "divided" into two ages: from 1993 to 2001, where the focus was mainly on mathematical and test-based models and, after 2001, where the focus was on techniques and models based in predicting quality requirements. More details about it can be seen in [Alvaro et al., 2005a].

By looking at the works covered in [Alvaro et al., 2005a], which represent the history and the current state-of-the-art in component certification, we may notice that this is a still immature area. Further research is needed in processes, methods, techniques, models, and tools, in order to obtain well-defined standards to software component certification.

In general, the main certification idea's is bringing quality to a certain software product, in this case software components. One of the core goals to achieve quality in component is to acquire reliability on it and, in this way, increase the component market adoption. Normally, the software component evaluation occurs through models that measure its quality. These models describe and organize the component quality characteristics that will be considered during the evaluation. So, to measure the quality of a software component it is necessary to develop a quality model. In this way, we aim to investigate a Component Quality Model (CQM), identifying its characteristics, the sub-characteristics, the quality attributes and the related metrics that compose the model.

3. A Component Quality Model (CQM)

In general, there is no consensus yet on how to define and categorize software component quality characteristics [Goulão and Abreu, 2002a], [Goulão and Abreu, 2002b], [Bertoa and Vallecillo, 2002]. Here we will try to follow as much as possible a standard terminology, in particular the one defined by ISO/IEC 9126 [ISO/IEC 9126, 2001].

The ISO/IEC 9126 is a generic software quality model and it can be applied to any software product by tailoring it to a specific purpose. The main drawback of the existing international standards is that they provide very general quality models and guidelines, and are very difficult to apply to specific domains such as Component-Off-The-Self (COTS) and Component-Based Software Development (CBSD).

The component quality model presented here (Table 1) is based on ISO/IEC 9126 and some adaptations for components were accomplished [Alvaro et al., 2005b]. The sub-characteristics, quality attributes, metrics and kind of metrics adopted could be seen at [Alvaro et al., 2005c].

Table 1. A Component Quality Model.

Characteristics	**Sub-Characteristics (*Runtime*)**	**Sub-Characteristics (*Life cycle*)**
Functionality	Accuracy Security	Suitability Interoperability Compliance Self-contained
Reliability	Fault Tolerance Recoverability	Maturity
Usability	Configurability	Understandability Learnability Operability
Efficiency	Time Behavior Resource Behavior Scalability	
Maintainability	Stability	Changeability Testability
Portability	Deployability	Replaceability Adaptability Reusability
Marketability	Development time Cost Time to market Targeted market Affordability	

Besides concentrating on quality characteristics only, we also created and added to the model other characteristics level called *Marketability*. This information is not important to evaluate the quality of a component, but is important to analyze some factors that bring credibility to the component customers.

Still on, we identified some characteristics that bring relevant information for new customers, such as *Productivity*, *Satisfaction*, *Security* and *Effectiveness*. According the ISO/IEC 9126, theses characteristics are called *Quality in Use* characteristics. This is the user's view (i.e. developer's) of the component, obtained when they use a certain component in an execution environment and analyze the results according their expectation. These characteristics show if the customer can trust in a component or not.

Finally, we identified some additional characteristics that are interesting to the component quality model (Table 2). These characteristics are called *Considerable Information* and are composed of: *Technical Information*, which is important to the customer analyze the actual state of the component (i.e. if the component has evolved, if any patterns was used in the implementation, which is the component version, amongst other information); and *Responsible*, which is interesting to the customer know who is the responsible for that component, i.e. who maintain that component.

Table 2. Additional Information.

Additional Information	Technical Information • Component Version • Programming Language • Patterns Usage • Lines of Code • Technical Support Organization Information • CMM Level • Organization's Reputation

4. A Preliminary Evaluation

In order to determine whether the CQM meets its proposed goals, a formal case study was performed. This section describes the steps of the formal case study and presents how these steps were performed in the study of this work.

The plan of the formal case study to be discussed follows the model proposed in [Wohlin et al., 2000] and the organization adopted in [Barros et al., 2002]. The definition and the planning steps to be presented in the following sections are described in the future tense, symbolizing the precedence of the plan in relation to its execution.

This formal case study was meant to be as close as possible to an empirical study. It contemplates important points that a good empirical study should have, such as the previous planning and the precise definition of the null and alternative hypotheses to be evaluated. However, it cannot be considered a complete empirical study because it does not contemplate essential parts of an empirical evaluation, such as [Basili et al., 1986]: Pilot Project, Qualitative Analysis, Internal/External Validity of the Study, Validity of the Construction of the Study, among other points.

4.1. Definition of the Formal Case Study

According to the Goal Question Metric Paradigm (GQM) [Basili et al., 1986], the main objective of this study is to:

Analyze *the current gap between the required and provided information of the main component marketplaces and a Brazilian software factory*
for the purpose *of evaluating the component quality model*
with respect *to the efficiency of the model*
from the point of view of the *researchers, software and quality engineers*
in the context of the *software component certification area.*

4.2. Planning of the Formal Case Study

Context. The objective of the study is to evaluate how much of the information required for the metrics in the proposed CQM is currently available from the most widely used component markets.

Object of Study. The objective of the study is evaluate the CQM usage, considering the software components of the most popular component markets from the Internet, such as

JARS[5], *Flashline* and *ComponentSource*, and the software components of a Brazilian software factory[6]. Thus, from these three sites, *JARS* and *Flashline* were finally discarded. First, *JARS* mainly offers Java applets, and the information provided for each product is very scarce, basically a pointer to the Web site of the Applet developer (which usually provided few more information). By the other hand, the *ComponentSource* marketplace acquired *Flashline's* marketplace in 2003 and all of its products incorporated into *ComponentSource's* offerings. Therefore, we decided to discard *Flashline* and concentrate only on *ComponentSource*, which covered both, and suddenly became the only large and widely used software component market.

Subject. The subject of the study will be a MSc. student at Federal University of Pernambuco and, system and quality engineers at the Brazilian software factory.

Instrumentation. The subject will act as a system and quality engineer, and will use Excel spread sheets in order to compute the component information found in the analyzed markets.

Criteria. The quality focus of the study demands criteria that evaluate the real efficiency of the model in measuring software component quality. This criteria will be evaluated quantitatively (**(i)amount of the component quality attributes found**, **(ii) amount of the component quality attributes measured** and **(iii) amount of the component information found in the market and not covered in the model**).

Hypothesis. An important aspect of formal case studies is to know and to formally state what is going to be evaluated in the formal case study. A set of hypotheses was selected, as described next.

- Null hypotheses, H_0: these are the hypotheses that the experimenter wants to reject strongly. In this study, the null hypotheses determine that the CQM is not efficient to measure the component quality information that is available in the component marketplaces and in a software factory. According to the selected criteria, the following hypotheses can be defined:

$H_{o'}$: amount of the component quality attributes of the model that were found in the market < 70%

$H_{o''}$: amount of the component quality attributes of the model that could be measured < 70%

$H_{o'''}$: amount of the component information that was found in the market and that is not covered in the model > 5%

The values of these hypotheses (70%, 70% and 5%, receptivity) were achieved through the feedback of some researchers of RiSE group and, software and quality engineers of a Brazilian software factory. Thus, these values constitute well-defined indices which the model must achieve in order to prove its viability.

- Alternative hypotheses: these are the hypotheses in favor of that which the null hypotheses reject. The formal case study aims to prove the alternative hypotheses by contradicting the null hypotheses, as follows:

H_1: amount of the component quality attributes of the model that were found in the market >= 70%

H_2: amount of the component quality attributes of the model that could be measured >= 70%

H_3: amount of the component information that was found in the market and that is not covered in the model <= 5%

[5] http://www.jars.com

[6] Currently, this company has about 700 employees and is in preparation to obtain the CMM level 3.

Validity of the Conclusion of the Study. The validation of the conclusion of the study measures the relation between the treatment and the result, and determines the capability of the study to generate conclusions [Wohlin et al., 2000].

4.3. Design of the Formal Case Study

Data Used in the Study. Fifty components were selected from *ComponentSource* (13 business components, 14 infra-structure components, 13 interface components and 10 database components). The components were chosen from *Best Sellers* and *Top Reviews* categories. The selection criterion was to choose only the most recommend and mostly used components. The idea was to evaluate the best components of the market, because such components would theoretically have more information that could be used to evaluate quality. However, this has proven not to be true.

From the Brazilian Software Factory, only six components were obtained. However, differently from those obtained from *ComponentSource*, which contained just a few samples and some documentation, these components were completely available (i.e. source code, documentation, models, samples, etc.).

Among a set of 13,000 components of *ComponentSource*, 50 components were selected and, by the other hand, among a set of 20 component of a Brazilian software factory, 06 components were selected. Thus, the data obtained from this study can be used to make comparisons between the markets because the data selected constitute approximately the same weight (50 from 13,000 and 06 from 20 components).

4.4. Instantiation of the Formal Case Study

Selection of the Subjects. For this study, no students were selected. Only one MSc. student participated in the formal case study.

Random Capability. The selection of the subjects for the formal case study was not random, since only one MSc. student accomplished the study.

Analysis Mechanism. To evaluate the hypotheses of the study, mechanisms of descriptive statistics (e.g. the mean), will be used.

4.5. Instantiation of the Formal Case Study

Realization. The formal case study was conducted as part of a MSc. Course in Computer Science, during first semester in 2005, at Federal University of Pernambuco. The formal study was executed over a period of 2 months, conducted by a single MSc. student. All the components were deployed, used and tested (through a set of examples developed and using demos available). The computer used has Windows operating system, 1.8GHz of processor speed and 512mb of RAM.

4.6. Analysis of the Formal Case Study

According to the required information of the CQM, each software component was analyzed together with its related assets in order to provide the information required.

Descriptive Analysis. Once the necessary information was collected, the analysis could be performed (Table 3). Thus, the percentages for null hypothesis H_0, were calculated as the percentages of attributes that were found in the market in relation to all attributes of

the model. The percentages for null hypothesis $H_{0''}$ were calculated as the mean of all percentages of measurability for each market. Finally, the 0% values for null hypothesis H0''' were observed during the execution of the study. The standard deviation was not considered due to the small number of selected components.

Table 3. Mean of the data from the study.

	Mean	
	ComponentSource	***Software Factory***
H0'	72,58%	83,87%
H0''	32,92%	75,27%
H0'''	0%	0%

As shown in Table 3, in the case of the Brazilian software factory, the results have shown that the null hypotheses were rejected, while in the case of *ComponentSource* marketplace, the second null hypothesis was not rejected.

(i) Amount of the component quality attributes found: The CQM contains 62 quality attributes. In the study, 45 quality attributes (72,58%) were obtained from the information found in the *ComponentSource* marketplace, while in the Brazilian software factory this number was 52 (83,87%). These results reject hypothesis $H_{0'}$, which validates alternative hypothesis ***H_1: amount of the component quality attributes of the model that were found in the market >= 70%.*** This implies that the quality attributes that compose the model is compatible with the component information found in the component market and in the repository of the software factory;

(ii) Amount of the component quality attributes measured: During the study accomplished in the components available from the Brazilian software factory, 75,27% of the quality attributes required to the CQM was measured. In the case of the Brazilian software factory, null hypothesis ***$H_{0''}$ was rejected, validating hypothesis H_2: amount of the component quality attributes of the model that could be measured >= 70%.*** However, only 32,92% of the information required to the CQM could be measured into the *ComponentSource* marketplace. Therefore, in the case of *ComponentSource*, the null hypothesis $H_{0''}$ was not rejected. This can be explained by the lack of information (source code, documents, models, tutorials, API's, information about interfaces and components, etc) that could be freely accessed without buying the component, which was the approach taken by this study. However, in a real certification environment, these information would be available, and this number would probably be close to the number achieved in the case of the Brazilian software factory (75,27%); and

(iii) Amount of the component information found in the market and not covered in the model: Among all the component information found in the *ComponentSource* marketplace and in the Brazilian software factory, everything was covered by the proposed component quality model, i.e. the study did not find any information that could not fit into some attribute of the model. In this sense, null hypothesis $H_{0'''}$ was rejected, validating hypothesis ***H_3: amount of the component information that was found in the market and that is not covered in the model <= 5%.*** This indicates that the component characteristics that are defined in the CQM comprise a good set of characteristics, being a good candidate for software components evaluation.

Conclusion. The results shown in Table 3 confirm the existing gap between the information required by the "theoretical" metrics defined in current component quality models, and the information that is currently supplied by software components vendors and software factories that develop components. Even in the best case (the Brazilian software factory), almost 25% of the attributes could not be measured, and almost 16% of the attributes were not even present.

Another relevant point that was observed during this study is the gap between the information available in "black-box" components (those selected from *ComponentSource*) and in "white-box" components (those selected from a software factory). Although in some degree this was caused because this study did not actually buy the analyzed components, and therefore did not have access to more documents, it was observed that the source code is an important asset to be considered in component certification and must be available for a real component certification environment.

In order to improve this situation, component vendors should improve the available information about the components (both in quality and quantity) in order to aid the measurement task. Another improvement that could provide some benefits is to better structure and organizes the information aiming to facilitate the automation of the assessment processes as much as possible. Source code should be provided in order to maximize the amount of attributes that can be measured.

Still on, the quality characteristics and quality attributes of the model deserve further analysis, in order to refine the model in future evaluations. One example is the mean of attributes of the model that could be measured with information from the market. In this study, the amount of attributes that could be measured was 32,92% and 75,27% (for *ComponentSource* and the Brazilian software factory, respectively). The number for *ComponentSource* is extremely low, indicating a large gap between the information from the model and information from the market.

However, this calculation considered the amount of attributes measured in relation to ALL attributes from the model, including those that could not be found, which implied that this mean would never be greater than the amount of attributes that could be found. For example, if 70% of the attributes were found in the market, and EVERY component presented information to measure 100% of these attributes, the calculated mean would still be 70%.

A more representative result could be obtained by excluding from this calculation the attributes that could not be found in ANY component. With this in mind, another hypothesis could be elaborated: ***H2´: AMONG THE ATTRIBUTES THAT COULD BE FOUND, the amount of attributes of the model that could be measured >= 70%.*** Recalculating the means for *ComponentSource* and the Brazilian software factory, these values arise to 45,36% and 89,75%, respectively, better reflecting the measurability of the attributes. However, the gap between the information from components obtained from *ComponentSource* and from the Brazilian software factory is still a large one.

5. Main Contributions

The main contributions of this work could be split on three main aspects: **(i)** the realization of a survey related to the state-of-the-art in software component certification research; **(ii)** the proposition of a Component Quality Model to evaluate the software

component quality; and **(iii)** the accomplishment of a formal case study, in order to evaluate the proposed component quality model.

- **A Survey on Software Component Certification.** The main research contributions found in the literature, from the 90's until today, were analyzed in order to understand how the software component certification area has evolved during this timeline. Through this study, it became possible to elaborate a well-defined component certification framework, aiming the development of a consistent and efficient component certification process [Alvaro et al., 2005a];
- **The Component Quality Model.** The survey showed that software component quality is important to the component market and a quality model for software component is really necessary. In order to supply this necessity, a quality model for software components was defined, analyzing which quality characteristics and attributes are adequate for components and which metrics are useful for measuring these attributes [Alvaro et al, 2005b], [Alvaro et al, 2005c], [Alvaro et al, 2006a]; and
- **A Formal Case Study.** In order to determine whether the quality model meets its proposed goals, a formal case study was performed. This study analyzed the viability of the proposed component quality model, identifying its main drawbacks. In this sense, a preliminary evaluation of the model was accomplished and future evaluations have been planned [Alvaro et al, 2006b].

6. Concluding Remarks and Future Directions

The growing use of commercial products in large systems makes evaluation and selection of appropriate products an increasingly essential activity. However, many organizations struggle in their attempts to select an appropriate product for use in CBSD, which is being used in a wide variety of application areas and the correct operations of the components are often critical for business success and, in some cases, human safety. In this way, assessment and evaluation of software components has become a compulsory and crucial part of any CBSD lifecycle.

In this sense, in order to properly enable the evaluation of software components, supplying the real necessities of the software component markets, a component quality model is strictly necessary. Thus, this work presented a Component Quality Model proposed and its related quality characteristics. Still on, a formal case study was defined, planned, executed and analyzed, showing the viability of the component quality model proposed.

As future work, we intend to update the CQM with the new ISO/IEC 9126, revised in 2005 [ISO/IEC 25000, 2005]. Our research group is working with the definition of a Software Component Maturity Model (SCMM). Based on the CQM proposed, the SCMM will be constituted of certification levels where the components could be certified. The idea is to develop a model in which the component could increase its level of reliability and quality as it evolutes.

References

Almeida, E. S., Alvaro, A., Lucrédio, D., Garcia, V. C., Meira, S. R. L. (2004). "RiSE Project: Towards a Robust Framework for Software Reuse", In: IEEE International Conference on Information Reuse and Integration (IRI), Las Vegas, USA, pp. 48-53.

Alvaro, A., Almeida, E.S. and Meira, S.L. (2005a). "Software Component Certification: A Survey", In: The 31st IEEE EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Component-Based Software Engineering (CBSE) Track, pp. 117-125.

Alvaro, A.; Almeida, E. S.; Meira, S. R. L. (2005b). "Towards a Component Quality Model", In: The 31st IEEE EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Work in Progress Session, Porto, Portugal, 2005.

Alvaro, A., Almeida, E.S., Meira, S.L. (2005c) "Quality Attributes for a Component Quality Model", In: 10th International Workshop on Component Oriented Programming (WCOP) in conjunction with the 19th ACM European Conference on Object Oriented Programming (ECCOP), Scotland.

Alvaro, A.; Almeida, E. S.; Meira, S. R. L. (2006a). "A Software Component Quality Model". Submitted to the 31st IEEE EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Croatia.

Alvaro, A.; Almeida, E. S.; Meira, S. R. L. (2006b). "A Software Component Quality Model". Submitted to the 30st Annual International Computer Software and Applications Conference (COMPSAC), EUA, Chicago.

Barros, M.O., Werner, C.M.L., Travassos, G.H. "An Experimental Study about Modeling Use and Simulation in support to the Software Project Management" (*in portuguese*), In: 16th Brazilian Symposium in Software Engineering, Brazil, (2002).

Bass, L.; Buhman, C.; Dorda, S.; Long, F.; Robert, J.; Seacord, R.; Wallnau, K. C. (2000) "Market Assessment of Component-Based Software Engineering, Software Engineering Institute (SEI)", In: Technical Report, Vol. I, May.

Basili, V.R., Selby, R., Hutchens, D. "Experimentation in Software Engineering", In: IEEE Transactions on Software Engineering, Vol. 12, No. 07, pp. 733-743, 1986.

Bertoa, M. and Vallecillo, A. (2002) "Quality Attributes for COTS Components", In the Proceedings of the 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE), Spain.

Crnkovic, I. (2001). "Component-based software engineering - new challenges in software development", In: Software Focus, Vol. 02, No. 04, pp. 27-33.

D'Souza, D. F.; Wills, A. C. (1999). "Objects, Components, and Frameworks with UML, The Catalysis Approach". Addison-Wesley, USA.

Goulao, M. and Abreu, F. B. (2002a). “Towards a Component Quality Model”, Work in Progress Session of the 28th IEEE Euromicro Conference, Dortmund, Germany.

Goulao, M. and Abreu, F. B. (2002b). “The Quest for Software Components Quality”, In: The 26th IEEE Annual International Computer Software and Applications Conference (COMPSAC), England, pp. 313-318.

Heineman, G. T.; Councill, W. T. (2001). “Component-Based Software Engineering: Putting the Pieces Together”, Addison-Wesley, USA.

Heineman, G. T.; Councill, W.T.; Flynt, J. S.; Mehta, A.; Speed, J. R.; Shaw, M. (2000). “Component-Based Software Engineering and the Issue of Trust”, In: The 22th IEEE International Conference on Software Engineering (ICSE), Canada, pp. 661-664.

ISSO/IEC 9126, “Information Technology – Product Quality – Part1: Quality Model”, International Standard ISO/IEC 9126, International Standard Organization, June, 2001.

ISO/IEC 25000, 2005, Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE, International Standard Organization, July, 2005.

Jacobson, I.; Griss, M.; Jonsson, P. (1997). “Software Reuse: Architecture, Process and Organization for Business Success”, Addison-Wesley, Longman.

Krueger, C.W. (1992). “Software Reuse”, In: ACM Computing Surveys, Vol. 24, No. 02, June, 131-183.

Mcllroy, M. D. (1968). “Mass Produced Software Components”, In: NATO Software Engineering Conference Report, Germany, pp. 79-85.

Morris, J., Lee, G., Parker, K., Bundell, G. A. and Lam, C. P. (2001). “Software Component Certification”. In: IEEE Computer, Vol. 34, No. 09, pp. 30-36.

Trass, V.; Hillegersberg, J. (2000) “The software component market on the Internet, current status and conditions for growth”, In: ACM Sigsoft Software Engineering Notes, Vol. 25, No. 01, pp. 114-117.

Voas, J. M. (1998). “Certifying Off-the-Shelf Software Components”, In: IEEE Computer, Vol. 31, No. 06, pp. 53-59.

Wallnau, K. C. (2003). “Volume III: A Technology for Predictable Assembly from Certifiable Components”. In: Software Engineering Institute (SEI), Technical Report, Vol. III, April.

Weber, K. C., Nascimento, C. J. (2002). “Brazilian Software Quality 2002”. In: The 24th IEEE International Conference on Software Engineering (ICSE), EUA, pp. 634-638.

Wohlin, C. and Runeson, P. (1994). “Certification of Software Components”, In: IEEE Transactions on Software Engineering, Vol. 20, No. 06, pp. 494-499.

Wohlin, C., Runeson, P., Host, M., Ohlsson, C., Regnell, B., Wesslén, A. “Experimenta-tion in Software Engineering: An Introduction”, in: Kluver Academic Publishers, (2000).