TOWARD A QUALITY MODEL FOR CBSE
Conceptual Model Proposal

María A. Reyes, Maryoly Ortega, María Pérez, Anna Grimán
Luis E. Mendoza and Kenyer Domínguez
Departamento de Procesos y Sistemas, Universidad Simón Bolívar, P.O. Box 89000, Caracas 1080-A, Venezuela

Keywords: Quality model, Component-based software engineering, Components.

Abstract: In this paper, which is part of a research in progress, we analyze the conceptual elements behind Component-Based Software Engineering (CBSE) and propose a model that will support its quality evaluation. The conceptual model proposed integrates the *product perspective*, a view that includes components and Component-Based Software (CBS), as well as the *process perspective*, a view that represents the component and CBS development life cycle. The model proposal was developed under a systemic approach that will allow for assessing and improving products and processes immersed in CBSE. Future actions include proposing metrics to operationalize the model and validate them through a case study. The model application will allow studying the behavior of each perspective and the relationships among them.

1 INTRODUCTION

As software engineering has developed, new techniques and tools have been employed to improve development processes. Likewise, engineering scope has been expanded and sub-disciplines have been produced that serve as a framework for the software development process. In this regard, Component-Based Software Engineering (CBSE) is focused on the design and construction of applications that use software components; it is oriented toward the acquisition of existing components, and takes into account the interdependence between components and these applications (Pressman, 2005; Clemente and Hernández, 2003). In this regard, CBSE shares common grounds with Service Oriented Architecture (SOA) in its interest in developing organizational relationships: develop partnerships between developers and users; develop key processes. Mutual support between the application generator and service provider communities is necessary (Anderson, 2007).

One of the CBSE processes is the development of components as reusable entities for systems that requires applying established methodologies. Another process is the CBS development, which includes maintenance and improvement of systems through personalization and replacement of components (Councill and Heineman, 2001; Crnkovic, 2003).

CBSE objectives allow us to identify two perspectives within CBSE itself: the *product perspective*, a view that includes software components and CBS, and the *process perspective*, a view that represents the component and CBS development life cycle. However, for applications developed in CBSE to succeed, it is necessary to consider quality aspects, oriented to the identified perspectives. In this regard, it is possible to obtain a model that includes characteristics and sub-characteristics for each perspective through a quality model that provides a set of guidelines aimed at specifying quality requirements and assessment.

In this paper we analyze the conceptual elements behind Component-Based Software Engineering (CBSE) and propose a model that will support its quality evaluation. By using a systemic approach, our proposal integrates the product perspective, which includes software components and CBS, and the process perspective that includes how software components and CBS are constructed.

The remainder of this paper is structured as follows. In section 2 we discuss the background for this research. Then, in section 3, we give a brief description of the methodology followed in this work. Afterwards, in section 4 we describe each sub-model and the integrated model. Finally, section 5 provides the conclusions and future works.

2 BACKGROUND

CBSE emerges as a reuse approach for software Systems and it is oriented toward the design and construction of systems using software components (Pressman, 2005). The ability to construct full solutions interconnected through interfaces created with components is one important factor to consider within CBSE.

For the purposes of this research, we define *component* as a software element subject to composition by third parties, which can be used by other software elements and conforms with a component model (SEI, 2000; Meyer, 2003).

Councill and Heineman (2001) point out that components exhibit a number of characteristics: those referred to the properties inherent in the component, which are basic characteristics that every component must have, and the desirable or advanced properties of the components, which are related to the different functions offered by the component. The desirable characteristics of a component may be present or not, but this not jeopardizes its functionality; however, if components have these characteristics, they can guarantee the quality of the component as a product.

CBS, in turn, comprises a number of self-contained component units. Components may have been developed in different languages, executed on different platforms and distributed throughout different geographical locations (Gao et al., 2003). CBS may be built with its own software components, which can be specifically developed for the software or with commercial off-the-shelf (COTS) acquired for this particular purpose.

Additional to the concepts related to CBSE and as a function to measure and assess component quality, some authors have proposed quality models to assess and select software components (Bertoa et al., 2006; Simão and Belchior, 2003; Rawashdeh and Matalkah, 2006; Carvallo, Franch and Quer, 2006; Andreou and Tziakouris, 2007; Carvallo et al., 2007).

Furthermore, quality models for CBS has been presented by several authors: Jasmine and Vasantha (2007) propose a model to measure quality of CBS products; the Sedigh-Ali and Paul (2001) model is focused on measuring quality of COTS-based systems; and finally, Grunske (2007) presents a generic framework for early prediction of CBS quality.

All these proposals stress quality assurance oriented to measuring the properties of the CBS product; they do not focus on quality of those aspects related to the life cycle of CBS development as a whole or consider quality assurance of software components at the same time.

In this regard, Grunske (2007) highlights the importance of proposing unified quality assessment models for CBSE, which include the component development life cycle and its inclusion into systems so that to have the proper management of systems on a large scale, and support technology development and transference to industrial applications.

3 METHODOLOGY

Since the aim of this research was establishing a series of concepts behind CBSE (a CBSE conceptual model), we considered that a methodology for ontology creation was suitable for our purpose. The main goal is to present conceptual elements that allow defining a quality model for CBSE. In fact, Noy and McGuinness (2001) define ontology as an explicit and formal description of concepts in a discourse domain. They propose the following seven steps for creating an ontology, which were useful in our research:

1. Determine the domain and scope of the ontology
2. Consider reusing existing ontologies
3. Enumerate important terms in the ontology
4. Define the classes and the class hierarchy
5. Define the properties of classes
6. Define class relations
7. Create instances

The scope of this paper does not include detailing each one of the proposed steps; however, the activities carried out in every step are mentioned: (1) elements common to each definition, were identified to find the relevant terms; (2) ontologies using a product-process approach were not found, therefore existing ontologies were not reused; (3) significant terms are presented in the sub-models, specifying their hierarchical order as well as the relations among them; (4) properties are not specified or instances are not created at this level, because this has still to be applied to a case study. The outputs of the reminding steps are described in the following sections.

4 SUB-MODELS INTEGRATION

The quality model for the CBSE proposed herein consists of five sub-models established for the product and process perspectives. The component and the CBS sub-models are defined within the product perspective. The sub-models for the CBS life cycle and for the component life cycle are defined for the process perspective. It is important to highlight that in the process perspective, the sub-models represent process maturity models, which will allow to measure quality in each process phase. Finally, we show the Component Model, which presents standards that allow us to precise a component, its documentation and implementation, and also is the backbone of a CBS (Gao et al., 2003), providing support for development, communication, evolution, composition and implementation The Component Model is related with the component and CBS sub-model.

4.1 Product Perspective Sub-model: Components

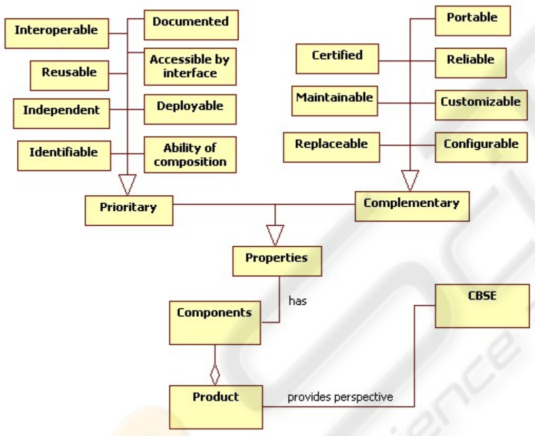

Figure 1: Component sub-model.

The relations established and the elements identified for the component sub-model consider the classifications proposed by Councill and Heineman (2001), Montilva et al., (2003), and Gao et al., (2003), who recognize the existence of mandatory characteristics related to identification, independence, composition, documentation and accessibility to the component through its interface. Concerning the desirable characteristics of a component, these authors refer to maintainability, personalization, portability, reliability, and certification. We classify characteristics as prioritary, those that must be met by every component effectively to ensure that the unit evaluated corresponds to a component, and complementary, those that add value to the component and allow us to certify the component quality. Figure 1 we show the characteristics defined in the component sub-model that will allow us to establish metrics to measure component quality.

4.2 Product Perspective Sub-model: CBS

The quality sub-model to measure those aspects related to CBS are based on the properties identified by Gao et al., (2003). These characteristics are unique and differ from traditional software systems, highlight the relations existing between the developed software and the components of which it consists; they also call for the need of component interoperability, and promote software and component reuse and the evolution (maintainability) of the developed systems, among others.

In Figure 2 we present the CBS quality model based in ISO/IEC 9126. It can be observed that the elements proposed correspond to the characteristics the CBS must comply with, which are related to components and their integration into the software.

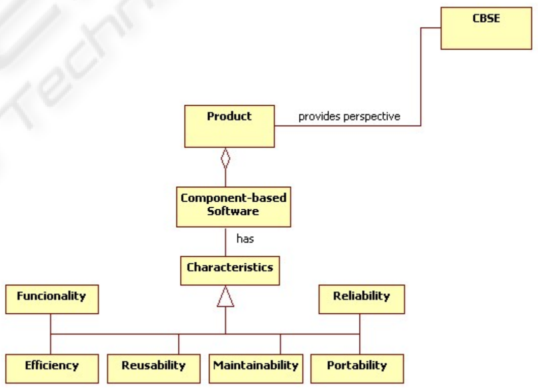

Figure 2: CBS sub-model.

4.3 Process Perspective Sub-model: CBS Life Cycle

Concerning the life cycle of CBS development, Cai et al. (2002), Crnkovic (2003), Sommerville (2005), and Pressman (2005) agree that CBS life cycle includes stages that are comparable to those in other processes; however, they consider that it is fundamental to include elements related to components and their composition into CBS. Figure 3 shows that CBS development life cycle includes seven stages that relate the software development life cycle to the elements associated to components and their composition.

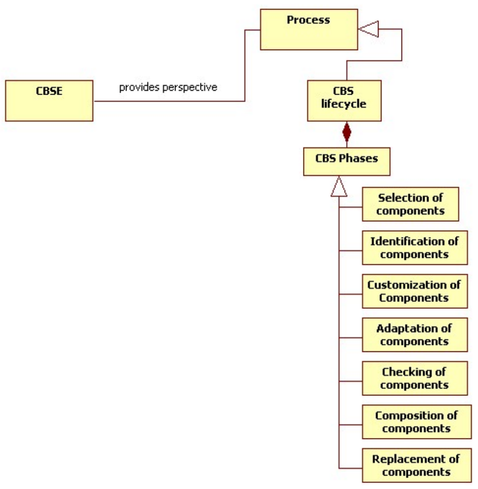

Figure 3: CBS life cycle sub-model.

4.4 Process Perspective Sub-model: Component Life Cycle

The component life cycle reflects how a component is constructed, its advantages and disadvantages, how it is integrated into a system, quality aspects and metrics required to assess process quality itself. Cai et al., (2002) and Lau and Wang (2006) identify three stages or phases in the life cycle: design, implementation and execution environment of the component. Figure 4 shows the phases identified for the component lifecycle: analysis of component requirements, development of the component within a system, and certification and personalization, viewed through the implementation and modification of the component for it to be adjusted to the requirements inherent in the system they are going to be integrated.

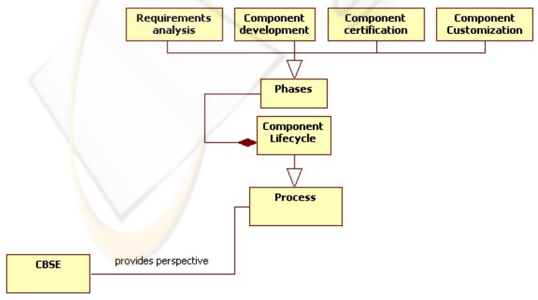

Figure 4: Component lifecycle sub-model.

4.5 Component Model

According to Sommerville (2005), a *components model* consists of a definition of the standards that allow for precising a component, its documentation and implementation. Besides, Gao et al. (2003) point out that the component model is the backbone of a CBS (e.g.: CORBA, DCOM, JavaBeans). Figure 5 presents the component model, which highlights the elements making up the components model, as well as the support for development, communication, evolution, composition and implementation of components. This components model is related with the CBS lifecycle in the composition phase, and with the component sub-model when the component complies the components model.

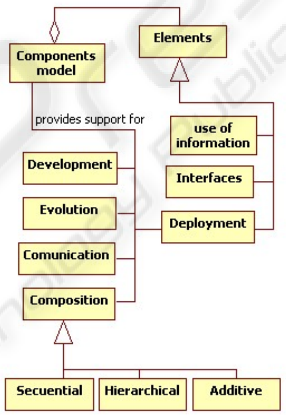

Figure 5: Component Model.

Once we followed the methodology for defining sub-models, we applied a method proposed by (Carvallo et al., 2004) to define and integrate quality models for CBSE. As a result, we obtained the conceptual model shown in Figure 6, which is based on integrated concepts and presents the relations not shown in the sub-models identified in CBSE. The relations between components and the component lifecycle, between CBS and its lifecycle; finally the relations between component models, components and CBS development are presented.

This global view of concepts also allows us to identify the important role of the sub-model named *components model* which depicts properties to be considered during the whole life cycle of the CBS. It is to be one of the main elements in any CBS quality evaluation model. This way, both perspectives are closely related to each other and that they are mutually affected. Notice that quality of a component directly influences CBS quality, through

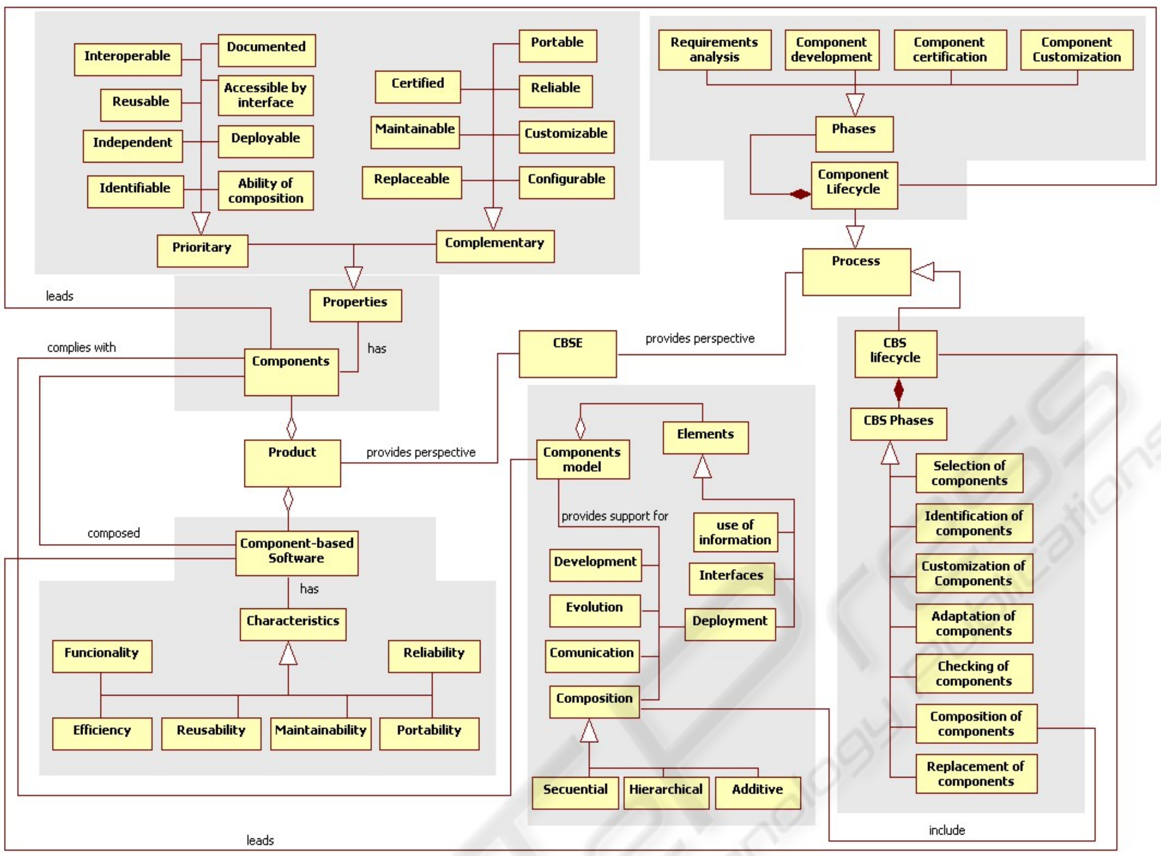

Figure 6: Integrated conceptual model for CBSE.

components models that are the backbone of these software systems. Furthermore, software construction, in each one of its phases, contemplates those component-related aspects that range from the component selection through their composition into systems and their maintenance or replacement.

5 CONCLUSIONS AND FUTURE WORK

We analyzed and presented a model that integrates the conceptual elements behind the formulation of a quality model for CBSE. From a systemic point of view, the model will allow us to assess and certify the product and process perspectives identified. Our proposal establishes and defines the relationships between products (components and CBS), and processes, related to how components and CBS are constructed. Furthermore, we can highlight two findings: 1) a close relationship exist between both identified perspectives: quality of a component directly influences CBS quality, 2) the component models are the backbone of these software systems.

Future works within the scope of this research-in-progress includes developing the metrics required to effectively measure all the aspects proposed in the integrated quality model. A GNU/Linux based distribution will be used for the case study on which the quality model is to be applied.

ACKNOWLEDGEMENTS

This research was partially supported by the National Fund of Science, Technology and Innovation, Venezuela (FONACIT), number G-2005000165 and project USB/DID S1-IN-CAI-01206.

REFERENCES

Anderson, W. 2007. What COTS and Software Reuse Teach us about SOA. In *Sixth International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems*. 141-149.

Andreou, A., Tziakouris, M., 2007. A quality framework for developing and evaluating original software components. In *Information and Software Technology*, 49, 122–141.

Bertoa, M. F., Troya, J. M. and Vallecillo, A., 2006. Measuring the usability of software components. In *Journal of Systems and Software*, 79(3), 427-439.

Cai, X., Lyu, M., Wong, K., 2002. Component-Based Embedded Software Engineering: Development Framework, Quality Assurance and a Generic assessment environment. In *International Journal of Software Engineering and Knowledge Engineering*, 12(2), 107-133.

Carvallo, J., Franch, X., Quer, C., 2006. Managing Non-Technical Requirements in COTS Components Selection. In *14th IEEE International Requirements Engineering Conference*, 316 - 321.

Carvallo, J., Franch, X., Quer, C., 2007. Determining Criteria for Selecting Software Components: Lessons Learned. In *IEEE Software*, 24(3), 84-94.

Carvallo, J., Franch, X., Grau, G., Quer, C., 2004. COSTUME: A Method for Building Quality Models for Composite COTS-based Software Systems. In *Fourth International Conference on Quality Software, Germany*. IEEE Computer Society Press, 214-223.

Clemente, P., Hernández, J., 2003. Aspect Component Based Software Engineering. In *Second AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software.*

Councill, W., Heineman, G., 2001. *Component-Based Software Engineering*. Addison Wesley.

Crnkovic, I., 2003. Component-based Software Engineering – New Challenges in Software Development. In *25th International Conference on Information Technology Interfaces.*

Gao, J., Tsao, J., Wu, Y., 2003. *Testing and Quality Assurance for Component-Based Software*. Artech House Publishers.

Grunske, L., 2007. Early quality prediction of component-based systems – A generic framework. In *The Journal of Systems and Software*, 80, 678–686.

Jasmine, K y Vasantha, R., 2007. DRE - A Quality Metric for Component based Software Products. In *Proceedings of World Academy of Science, Engineering and Technology*, 23, 380-383. Lau, K., Wang, Z., 2006. *A Survey of Software Component Models*. University of Manchester: Computer Science.

Meyer, B., 2003. The Grand Challenge of Trusted Components. In *25th International Conference on Software Engineering*.

Montilva, J., Arapé, N., Colmenares, J., 2003. Desarrollo de Software Basado en Componentes. In *IV Congreso de Automatización y Control, Mérida*.

Noy, N. y McGuinness, D., 2001. *Ontology Development 101: A Guide to Creating Your First Ontology.* Stanford University: Stanford, CA.

Pressman, R., 2005. *Software Engineering: A Practitioner's Approach,*. McGraw Hill. 6th. edition.

Rawashdeh, A. y Matalkah, B., 2006. A New Software Quality Model for Evaluating COTS Components. In *Journal of Computer Science* 2(4), 373-381.

Sedigh-Ali, S., y Paul, R., 2001. A Software Engineering Metrics for COTS-Based Systems. In *Computer*, 44-50.

SEI, Software Engineering Institute, 2000. *Volume II: Technical Concepts of Component-Based Software Engineering*. Carnegie Mellon University. 2nd edition.

Simão, R, Belchior, A. 2003. Quality characteristics for software components: Hierarchy and quality guides. In *Lecture Notes in Computer Science,* 2693, 184-206.

Sommerville, I., 2005. *Software Engineering*. Addison Wesley. 7th edition.

Szyperski, C., 2002. *Component Software Beyond Object-Oriented Programming*. Addison Wesley. 2nd edition.