A Quality Model for Embedded Software Component

Fernando Carvalho¹², Silvio R.L. Meira¹, Bruno Freitas², Nathalia Carvalho²

¹CIn – UFPE - CEP 50740-540 – Recife, PE – Brazil - +5581-2126.8430
²Faculty of Science and Technology of Caruaru – UPE - Caruaru, PE – Brazil
{ffc,silvio}@cin.ufpe.br, {nathalia.ellen,bruno.freitas}@upe.br

Abstract. Component-based software development is becoming more
generalized, representing a considerable efficient design for the embedded
industry. The perspective of reduced development costs and shorter life cycles
acts as a motivation for this expansion. However, several technical issues
remain unsolved before embedded software component’s industry reaches the
maturity exhibited by other hardware component industries. In this way, we
aim to propose an embedded component quality model, describing consistent
characteristics, quality attributes and related metrics for the components
evaluation. A preliminary evaluation to analyze the results of using the
embedded software component quality model proposed is also presented.

1. Introduction

One of the most compelling reasons for adopting component-based approaches to
embedded software development 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].

Most of the research dedicated to embedded software components is focused on
their functional aspects (i.e. component specification, development, tests, etc.). In our
ongoing research, we are concerned with the evaluation of embedded software
components quality, which important researchers on software components, such as
Wallnau [Wallnau, 2003], Heineman [Heineman & Councill, 2001], among others,
points that component certification is the future of the software components area. This
evaluation should be performed using a component quality model, which specifies the
quality characteristics that the software component has to attend. However, there are
several difficulties in the development of such a model, such as: (i) which quality
characteristics should be considered, (ii) how we can evaluate them and (iii) who should
be responsible for such evaluation [Goulão & Abreu, 2002].

The embedded industry whose aim is to construct systems by integrating
components will not be able to meet their objectives if they cannot find sufficient
number of components and component versions that satisfy certain functional and
quality requirements. Without a quality level, the component reuse may have
catastrophic results [Jezequel & Meyer, 1997].

Thus, the components reuse quality problems must be resolved to increase the
reliability, and third-party certification programs would help acquire trust in the reuse-
2. Towards an Embedded Software Component Quality Model

In general, there is no consensus yet on how to define and categorize software component quality characteristics [Goulão & Abreu, 2002], [Bertoa & Vallecillo, 2002], [Goulão et al., 2002]. The ISO/IEC 25000 [ISO/IEC 25000, 2005] 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 embedded design and Component-Based Software Development (CBSD).

After analyzing a set of models [Bertoa & Vallecillo, 2002], [Goulão & Abreu, 2002], [Simão & Belchior, 2003], [Torchiano et al., 2002] and [ISO/IEC 25000, 2005], an Embedded software component Quality Model (EQM) was proposed. The model is based on ISO/IEC 25000, contains marketing characteristics and some relevant component information’s which is not supported in other component quality models. The EQM proposed is composed of seven characteristics, as follows: Functionality; Reliability; Usability; Efficiency; Maintainability; Portability; and Marketability.

Although the model is proposed following the ISO/IEC 25000 standard, some changes were made in order to develop a consistent model to evaluate software components in embedded context: (i) the characteristics that were identified as relevant to the component and embedded context were maintained; (ii) one characteristic proved to be not interesting to evaluate embedded software components, and was eliminated; (iii) the name of one of the characteristics was changed in order to adequate it to the embedded component context; (iv) another level of characteristics was added, containing relevant marketing information for a software component certification process; and (v) some characteristics that complement the EQM with important component information were established.

2.1. Changes in relation to ISO/IEC 25000

Table 1 summarizes the changes that were performed in relation to ISO/IEC 25000. The characteristics and sub-characteristics that are represented in bold were not present in ISO/IEC 25000. They were added due to the need for evaluating certain CBSD-related properties that were not covered on ISO/IEC 25000. The sub-characteristic that is crossed was present in ISO/IEC 25000, but was removed in the proposed model. And, the sub-characteristic in italics had its name changed. Additionally, at the moment in which a characteristic can be observed or measured also allows establishing a classification. The characteristics can be observable at runtime (that are discernable at component execution time) and observable during the product life-cycle (that are discernable at component and CBSD).

Table 1. An Embedded software component Quality Model - EQM
Functionality
Real-time
Accuracy
Security

Suitability
Interoperability
Compliance
Self-contained

Efficiency
Time-Behavior
Resource behavior
Scalability
Energy consumption
Memory utilization

Reliability
Recoverability
Fault Tolerance
Safety

Maintainability
Analyzability
Stability
Changeability
Testability

Usability
Configurability
Understandability
Learnability
Operability

Portability
Deployability
Replaceability
Flexibility
Reusability

The Real-time sub-characteristic is essential to determine the correctness functionality the embedded component. In embedded domain practically all the systems had real-time requirements. The Self-contained sub-characteristic is intrinsic of a component and must be analyzed. The Safety is very important for reliability and dependable embedded systems, so this sub-characteristic is fundamental to compose the quality model. Additionally, the Configurability become essential to the developer analyze if the component can be easily configured. Thus, the developer verify the ability of configure a component in order to determine the complexity to deploy the component into a certain context. The Energy consumption is crucial for portable, mobile, autonomy system and others. It is essential to determine the project viability, battery size, system autonomy and others. The Memory utilization sub-characteristics gain emphasis in embedded domain because the resources such as code and data memory are very limited like code and data memory. Still on, the reason that embedded industry has adopted component-based approaches to software development is the premise of reuse. Thus, the Reusability sub-characteristics are so important to be considered too.

Additionally, some sub-characteristics were removed (crossed out) in order to adequate the model to the embedded component context, such as: Suitability, Interoperability, Understandability, Learnability, and Operability. These sub-characteristics are not present in the survey and they were judged to be of minor priority. In the Maintainability characteristic, the Analyzability sub-characteristic disappeared. In fact, practical experience has shown that components do not have Analyzability characteristics [Bertoa & Vallecillo, 2002].

Concurrently, some sub-characteristics had their names changed, as well as their meaning in this new context. The Installability, which in the proposed model has the new name of Deployability. After being developed, the components are deployed (not installed) in an execution environment to make their usage possible by other component-based applications that will be further developed. The Time behavior sub-characteristics had its name changed to Real-Time and it is part of functionality characteristics, because the time behavior in embedded systems is related to its correctness. Another characteristic that changed its meaning was Usability. Thus, the usability of a component should be interpreted as its ability to be used by the application developer when constructing a software product or a system with it. Finally, Adaptability was the last characteristic that changed names. Now, it’s called...
Flexibility. It indicates whether the component can be adapted or flexible to different specified environments. More information about those characteristics could be seen in [Alvaro et al., 2005].

Besides concentrating on quality characteristics only, we also created other characteristics level called Marketability. This characteristic presents some sub-characteristics that we think important to a certification process, such as: Development time; Cost: the cost of the component; time to market; targeted market; and affordability.

2.2. Component Quality Attributes

According to [ISO/IEC 25000, 2005], a quality model consists of four elements: (i) characteristics, (ii) sub-characteristics, (iii) attributes and (iv) metrics. A quality characteristic is a set of properties of a software product through which its quality can be described and evaluated. A characteristic may be refined into multiple levels of sub-characteristic. An attribute is a measurable physical or abstract property of an entity. A metric defines the measurement method and the measurement scale. The measurement process consists in assigning a number or category to an attribute, according to the type of metric that is associated to that attribute [ISO/IEC 25000, 2005].

Table 2. Embedded software component quality attributes.

<table>
<thead>
<tr>
<th>Character.</th>
<th>Sub-Charact.</th>
<th>Attributes</th>
<th>Character.</th>
<th>Sub-Charact.</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>Real-time</td>
<td>Response time</td>
<td>Resource Behavior</td>
<td>peripheral utilization</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Execution time</td>
<td>Energy consumption</td>
<td>Mechanism availability</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Worst case execution time</td>
<td>Data Memory utilization</td>
<td>Mechanism availability</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dead line</td>
<td>Program Memory utilization</td>
<td>Mechanism availability</td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td>Correctness</td>
<td>Data Encryption</td>
<td>Stability</td>
<td>Modifiability</td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td>Controllability</td>
<td>Certification</td>
<td>Changeability</td>
<td>Extensibility</td>
<td></td>
</tr>
<tr>
<td>Audibility</td>
<td>Standardization</td>
<td>Dependability</td>
<td>Customizability</td>
<td>Modularity</td>
<td></td>
</tr>
<tr>
<td>Compliance</td>
<td>Certification</td>
<td>Error Handling</td>
<td>Testability</td>
<td>Extensive component test cases</td>
<td></td>
</tr>
<tr>
<td>Self-contained</td>
<td>Dependability</td>
<td>Deployability</td>
<td>Test suite provided</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recoverability</td>
<td>Mechanism availability</td>
<td>Mechanism efficiency</td>
<td>Deployability</td>
<td>Complexity level</td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td>Safety</td>
<td>Environment analyze</td>
<td>Replaceability</td>
<td>Backward Compatibility</td>
<td></td>
</tr>
<tr>
<td>Fault Tolerance</td>
<td>Integrity</td>
<td>Flexibility</td>
<td>Mobility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td>Configurability</td>
<td>Understandability</td>
<td>Reusability</td>
<td>Architecture compatibility</td>
<td></td>
</tr>
</tbody>
</table>
of rules for the interpretation of the measurement data. The resulting measurement model has three levels:

1. Conceptual level - **GOAL**: A goal is defined measurement for Products, Processes and Resources.

2. Operational level - **QUESTION**: A set of questions is used to characterize the way the assessment/achievement of a specific goal is going to be performed based on some characterizing model, with respect to a selected quality issue and to determine its quality from the selected viewpoint; and

3. Quantitative level - **METRIC**: A set of data is associated with every question in order to answer it in a quantitative way. (Objective and Subjective).

An example of metrics usage will be presented. However, it is important to highlight that those metrics must be defined in an embedded component evaluation context. Those ones presented here are only to guide the evaluation team during the definitions of the metrics in the component’s first evaluation process. For example, for Accuracy Sub-Characteristic the following metric could be applied:

**Table 3. A metric example**

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Question</th>
<th>Metric</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sub-Characteristic</strong></td>
<td>Accuracy</td>
<td></td>
<td>Based on the amount of tests executed, how much test results return with precision?</td>
</tr>
<tr>
<td>Quality Attribute</td>
<td>Correctness</td>
<td>Precision on results / Amount of tests</td>
<td>0 &lt;= x &lt;= 1; which closer to 1 is better</td>
</tr>
<tr>
<td>Goal</td>
<td>Evaluates the percentage of the results that were obtained with precision</td>
<td>Precision on results / Amount of tests</td>
<td>0 &lt;= x &lt;= 1; which closer to 1 is better</td>
</tr>
</tbody>
</table>

An interesting aspect here is that the evaluation team defines the Interpretation of the metrics definition. Thus, the collection and analysis of those metrics becomes more feasible, repeatable, reproducible and easier to understand the results in a way that the whole team knows how the attribute was collected.

**2.3. Other relevant Component Information**

In an effective software component certification process, some kind of information is needed in order information. Besides the characteristics presented early, we identified other Additional Information. These additional characteristics are composed of: Technical Information (Component Version, Programming Language, Patterns Usage, Lines of Code and Technical Support) and Organization Information (CMMI Level and Organization’s Reputation).

Technical Information is important for developers analyze the current state of the component (i.e. if the component has evolved, if any patterns were used in the implementation, which kinds of technical support are available, etc.). Besides, it is interesting to the customer that he knows who is the responsible for that component (e.g. components developed by a CMMI level 5 company probably is more reliable than a component create by an unknown embedded design). Thus, it is identified the necessity of the Organization Information.

**3. Concluding Remarks and Future Directions**

This work proposed an initial embedded software component quality model in order to establish the requirements to a well-defined embedded software component certification
process. Our research group, in conjunction with the industry, aim to investigate the embedded software component certification area in order to: (i) establish a well-defined embedded software component quality model; (ii) define a framework (and corresponding metrics) to track the components properties; (iii) build a certification method and, finally, (iv) developed a structured embedded component certification process.

The long term plan is, clearly, to achieve a degree of maturity that could be used as an embedded software component certification standard for embedded industry, making it possible to create a Component Certification Center. Currently, our research group is working with the definition of an Embedded software component Maturity Model (EMM). Based on the Embedded software component Quality Model (EQM) proposed, the EMM will be constituted of certification levels where the components could be certified. The intention is to develop a model in which the component could increase its level of reliability and quality as it evolves (the EMM is based on the same CMM principles [Paulk et al., 1993]).

4. References