Towards a Components Quality Model

Miguel Goulão
Information Systems Group (INESC)
FCT/UNL
Departamento de Informática
2825-114 Monte da Caparica, Portugal
+351 212948536 (ext. 10731)
miguel.goulao@di.fct.unl.pt

Fernando Brito e Abreu
Information Systems Group (INESC)
FCT/UNL
Departamento de Informática
2825-114 Monte da Caparica, Portugal
+351 212948536 (ext. 10707)
fba@di.fct.unl.pt

1. Introduction

Component Based Development (CBD) is becoming increasingly important for the software industry. According to studies performed by four major market researchers (Gartner Group, Gica, Ovum and PriceWaterhouseCoopers) the component market has been growing steadily over the recent years and forecasts for its growth are made [1] [2].

CBD is supposed to reduce the cost and time to market of software applications while increasing their quality. Since components are reused in several occasions, they are likely to be more reliable than software developed from scratch, as they were tested under a larger variety of conditions. Cost and time savings result from the effort that would otherwise be necessary to develop and integrate the functionalities provided by the components in each new software application.

Most of the research dedicated to software components is focused on their functional aspects. In our ongoing research, we are concerned with the evaluation of software components quality. This evaluation should be performed using a component quality model. There are several difficulties in the development of such a model, such as (1) which quality characteristics should be considered, (2) how we can evaluate them and (3) who should be responsible for such evaluation.

The development and validation of a model that answers these questions would offer a very useful evaluation tool to clients looking for components to include in their software applications. Currently, little or no information on quality is provided by component vendors. For the sake of discussion, we will use here the following component definition: a software component is an independently deployable implementation of some functionality, to be reused as-is in a broad spectrum of applications[1].

2. A Quality Model for Components

2.1 Choosing Quality Characteristics

There are two main threads in the research of software quality: one of them is concerned with evaluating the software process quality while the other focuses on the product quality itself. Our current research deals with the latter.

Several product quality models can be found in the literature, such as [3-9]. Among these models, the most consensual model is the one offered by ISO9126. These models include generic software quality attributes. Besides, they were conceived at the system level, not at the component one. While some of their characteristics are appropriate to the evaluation of software components, others are not well suited for that task. Some of their quality characteristics, such as fault tolerance are typically evaluated at the system level, rather than for each of the components.

A quality model for components should be tailored to use only the characteristics that apply to components. In [10] one such model is proposed for the evaluation of COTS components. The model proposed in this paper has a wider spectrum of application, as it is targeted for software components in general. As a starting point, we will use the quality model described in [9] (see Table 1) which is an attempt to improve some of the shortcomings of the ISO9126 model. We will adapt it to the special needs of quality components.

We are currently researching which of these characteristics are applicable to software components. Discrimination between white box and black box characteristics will be made. The former are more suited for component quality evaluation on the to the developers of components point of view, while the latter are of special interest to the components consumers (developers using components in the construction of their applications).

Characteristics	Sub-characteristics
Functionality	Suitability, Correctness, Interoperability, Security, Compliance
Reliability	Maturity, Fault Tolerance, Recoverability, Compliance
Usability	Installability, Learnability, Configurability, Operability, Monitorability, Compliance
Efficiency	Time behaviour, Resource behaviour, Compliance
Maintainability	Diagnoseability, Repairability, Extensibility, Testability, Compliance
Portability	Adaptability, Replaceability, Compliance

Table 1 - Quality Characteristics

2.2 Evaluation of Characteristics

The evaluation of quality characteristics should be repeatable: different evaluation teams in different evaluation sites should obtain the same evaluation for a given software component. In order to achieve this, quality characteristics have to be evaluated through objective well-defined metrics. To avoid misinterpretations of metrics definitions, these metrics should be formally defined. We plan to use OCL [11] for this formal definition. OCL provides a good balance between formality and understandability. Since it is a part of UML, it is a suitable language for this formalization, as it can be easily understood by software engineers versed in the OO and CBD paradigms. Other formalization techniques require background knowledge that is not held by the majority of the software engineering community.

As with component quality characteristics, we will also use a classification criterion for metrics: whether they are white box or black box metrics. This classification is relevant for the definition of responsibilities in the evaluation process. The software component quality model should not be tied to a particular metrics set. In fact, there is no metrics set that is commonly accepted as the standard metrics set for software components. We can, however, map our model to a metrics set to illustrate how this mapping can be achieved.

2.3 Responsibility for the Evaluation

Ideally, evaluation should be performed by a trusted third party independent evaluator, as this is essential for component certification, a long term objective of this work. However, this creates a few problems, as components are distributed as black boxes, thus making the usage of white box certification techniques unavailable. We will research this topic using as a starting point other attempts to deal with it, such as [12].

3. Conclusions and Future Work

Mechanisms to provide trust to component users are a common market practice enforced by component purchasers and often a legal imposition in other industries. Component certification is such a trust mechanism. As a long term goal, we aim to contribute to the development of a certification process that is adequate to software components. The first step towards that objective is the definition of a quality model that can be used as a framework in component evaluation. We are currently working towards this goal.

References

[1] L. Bass, C. Buhman, S. Comella-Dorda, F. Long, J. Robert, R. Seacord, and K. Wallnau, "Volume I: Market Assessment of Component-Based Software Engineering," Software Engineering Institute, Technical Note CMU/SEI-2001-TN-007, May, 2001.

[2] J. D. Williams, "Raising Components," Application Development Trends, vol. 7, pp. 27-32, 2000.

[3] ISO9126, "Information Technology - Software Product Evaluation - Software Quality Characteristics and Metrics," . Geneva, Switzerland: International Organization for Standardization.

[4] J. McCall, "Quality Factors," in Encyclopedia of Software Engineering, vol. I+II, J. J. Marciniak, Ed.: John Wiley & Sons, 1994, pp. 958-ff.

[5] B. W. Boehm, B. K. Clark, E. Horowitz, R. Maduchy, R. Selby, and C. Westland, "An overview of the COCOMO 2.0 software cost model," , 1995.

[6] L. J. Arthur, Measuring Programmer Productivity and Software Quality: Wiley-Interscience, 1985.

[7] R. B. Grady and D. L. Caswell, Software Metrics: Establishing a Company-Wide Program. Englewood Cliffs, NJ, EUA: Prentice-Hall, 1987.

[8] B. Meyer, Object-Oriented Software Construction, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1997.

[9] F. B. Abreu, "Comparing Software Quality Models," : INESC Technical Report (in Portuguese), 2001.

[10] M. Bertoa and A. Vallecillo, "Quality Attributes for COTS Components," presented at 6th International Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'2002), Málaga, Spain, 2002.

[11] OMG, "Object Constraint Language Specification (version 1.1),", Rational, Ed. Menlo Park, CA, EUA: Object Management Group, 1997.

[12] B. Councill, "Third-Party Certification and Its Required Elements," presented at 4th ICSE Workshop on Component-Based Software Engineering, Toronto, Canada, 2001.