On the Use of Quality Models for COTS Evaluation

Juan P. Carvallo, Xavier Franch, Gemma Grau, Carme Quer
Universitat Politècnica de Catalunya (UPC)
c/ Jordi Girona 1-3 (Campus Nord, C6) E-08034 Barcelona (Catalunya, Spain)
{carvallo, franch, ggrau, cquer}@lsi.upc.es

Abstract

Software quality models have been widely used for assessing software quality during software systems development, but their use for facilitating COTS evaluation (as a basis for COTS selection) has been not so extensively reported. We think that this situation should change, because quality models present some interesting properties for driving COTS evaluation: they are highly structured; there exist some standards available; they can be tailored to specific domains; they provide a measurable basis for the evaluation of software quality; etc. In this paper, we provide a summary of the experiences of our group in the use of quality models for COTS selection, and we highlight some open issues that we are currently dealing with.

1. Introduction

A quality model is "the set of characteristics and the relationships between them which provide the basis for specifying quality requirements and evaluating quality" [1]. In the field of COTS-based systems, a quality model for a given COTS domain provides a taxonomy of software quality factors that are of interest for this domain and also metrics for computing their value. Once available, requirements over the domain may be stated with respect to the quality model. COTS components quality can be evaluated and their behavior can be compared with requirements based on a common description.

The goal of this experience report is to disseminate the key points of the knowledge of our research group, GESSI (in Catalan, *Group of Software Engineering for Information Systems*) on the use of quality models for COTS evaluation, and to identify some open points subject of further investigation.

2. Quality Models in GESSI: Current State

The ISO/IEC 9216 as quality framework

A great deal of the existing quality models are ad-hoc proposals build to be used in particular contexts and applications; thus, their reusability is compromised. Not only quality elements are context-dependent, but also all the quality concepts included. The use of widespread standards is thus highly recommended. Elements and concepts of these models are likely to be reused because they are the result of the consensus of large communities.

Among the available standards, we have selected the ISO/IEC 9126-1 for the following reasons: it just fixes some general characteristics, and so the quality model may be tailored to any specific software domain; it explicitly recognizes the convenience of creating hierarchies of quality features, which is essential in order to build structured quality models; it is widespread; it has been recently updated, although in fact a new standard (ISO/IEC 25020, known as SQuaRE) is being developed putting together some quality-related standards.

In our work we made assumptions for some concepts in the standard that are not clearly defined: characteristics are non-measurable and are included in quality models with a classification aim; metrics may be subjective or objective, depending on if it is possible or not to establish a precise measurement procedure; subcharacteristics and derived attributes are measurable and may be decomposed into other subcharacteristics or into attributes; basic attributes are objectively measurable and may not be further decomposed; and, attributes may be related with more than one quality factor in the upper level of the hierarchy, having a different metric in every case. In [4] you can find the UML class diagrams that describe the framework and that were used as a base for the development of QM.

More comprehensive models to embrace knowledge

The use of the ISO/IEC 9126-1 quality model as starting point raises some problems. On the one hand, the usual one when using standards: in some contexts it is not exactly what is wanted. But we have found other types of problems. First, it is ambiguous at some points, see [2]. Second, its coordination with non-technical issues (e.g., vendor-related) when used during selection is not easy. Third, relationships among quality factors are not recorded.

To overcome these difficulties, we have proceeded the following way. First, we have defined a rigorous data model in the form of a UML class diagram stating precisely the meaning of all the quality model elements. Second, we have included in the model (although clearly

separated) non-technical issues in the same layout than quality ones (this is allowed by the standard provided that new issues are explicitly linked with existing ones). Third, we have used some relationship categories (e.g., conflict, improves, collaborate, etc.) and we propose to include them as an informative annex of the quality model.

Our IQMC as construction process

In large COTS domains, building quality models can be a time-consuming and cumbersome task. Therefore, a process to guide the construction is required. Unfortunately, neither the ISO/IEC standard nor other approaches propose this kind of processes. As a result, we have defined a method of our own, the IQMC (Individual Quality Model Construction) method [3].

IQMC uses the quality model presented in the ISO/IEC standard as starting point and then applies 7 steps (intertwined and iterated as needed), that can be summarized as: analyze the COTS domain; refine the quality model hierarchy; state the relationships of the resulting quality factors; define metrics for the quality factors.

We remark the iterative and intertwined nature of IQMC. On the one hand, model elements are identified as knowledge of the domain increases, thus the model may be extended and refined all the way through the process. On the other hand, the level of detail to which the quality model has to be constructed depends on the context of use and the final application for which the quality model is intended.

A taxonomy of COTS domains to improve reuse

In our practices, we have observed that some quality factors are not bound to individual COTS domains but to some of them, which form a kind of category. This fact yields to the believe that reuse can be done better than just copying-and-pasting quality factors. The key point is that the quality factors that COTS domains have in common can be inherited from a more generic quality model; therefore, new quality models can be build not starting from the scratch but from a first version resulting from the inheritance of its ancestor quality models. This classification is used to build a taxonomy of COTS categories and domains that is used both during the definition of quality models and the selection of COTS components.

We have proposed a taxonomy-building approach based on decision trees theory [4]. We use appropriate classification attributes to structure the domains, and then we bound quality factor hierarchies at every node. Also, relationships among these domains may be explicitly declared, to keep track of how particular COTS components may depend on others; this information is very valuable when evaluating and selecting components.

DesCOTS as tool support

Tool support has been considered necessary from the very beginning of our experiences. If not, the complexity itself of the construction process can be very difficult to handle. Furthermore, reuse is only feasible within this assumption.

After considering some existing alternatives, we have finally decided to develop our own support environment, DesCOTS (*Description, evaluation and selection of COTS components*). The system embraces different subsystems (see fig. 1 for an *i** SD actor-based description) that can operate as separate tools: the *Quality Model Tool* allows to define quality models; the *COTS Evaluation Tool* allows the evaluation of components; the *COTS Selection Tool* allows the definition of requirements that drive the COTS component selection; and *the Taxonomy Tool* allows to organize COTS domains as a taxonomy supporting reuse of quality models. Fig. 2 provides a snapshot of the Quality Model Tool.

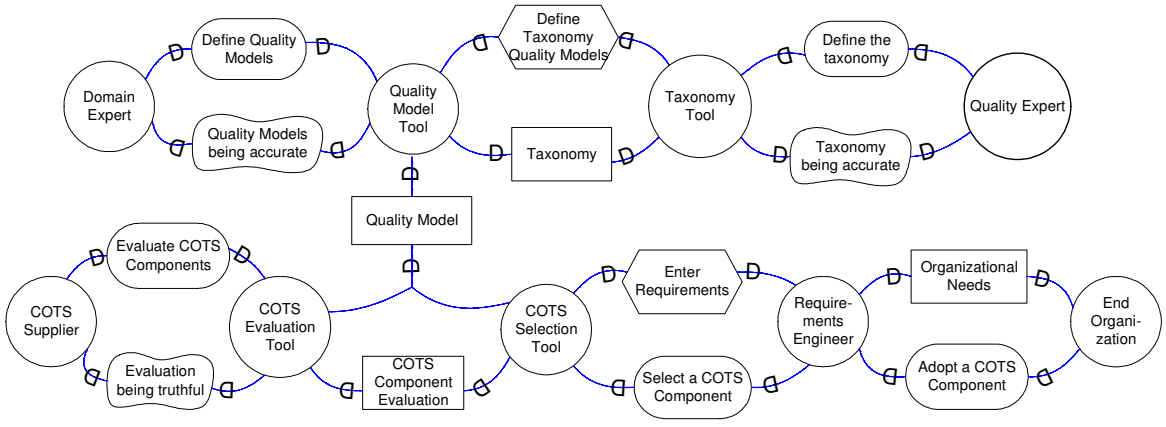

Fig. 1: Actor-based model for the DesCOTS system

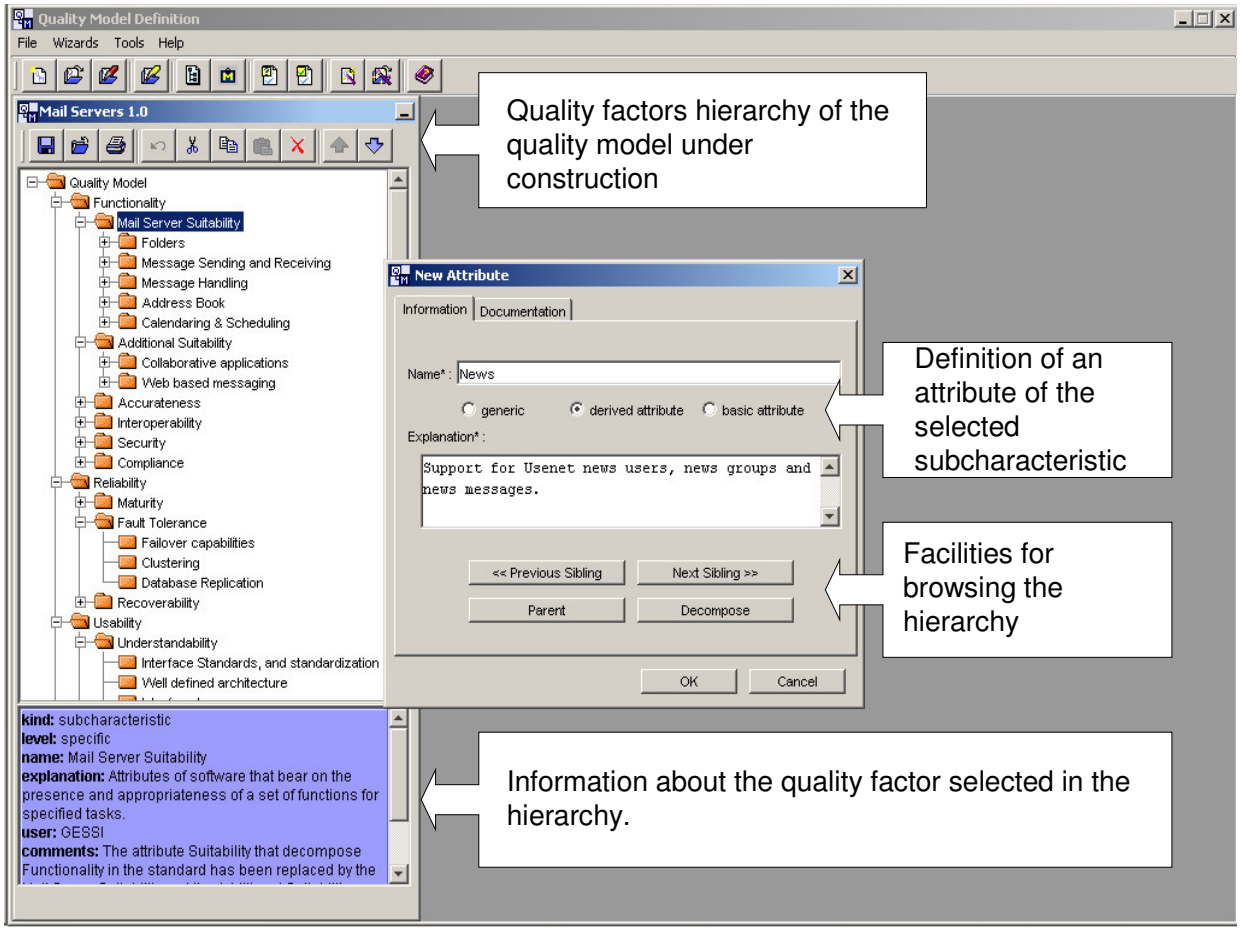

Fig. 2. Quality Moel Tool snapshot: adding a new attribute to the quality model hyerarchy

The DesCOTS system was designed with the purpose of being used by different kind of users that will probably not be working in the same location. To allow data interchange between these different users, we decided to design the system following a client/server architectural pattern. Each tool has its client program which can be installed independently from the others and just needs to access the server program to get and store the shared data using HTTP/XML, JAVATM servlets and MySQL. All the libraries used are open source, following the aim of getting openness and flexibility on the system development and distribution. The client parts of the tools are small applications of 20 Mb on average size, plus 35Mb of the Java Virtual Machine (JVMTM) so it is easy to download and distribute. They run on a low-profile hardware and are easy to install.

Rigorous models for representing knowledge

Construction of software quality models has to be supported by a deep exploration of the domain of interest.

The results of this study cannot entirely rely on common sense and informal documents. Regardless of the level of detail and precision used to describe the domain in natural language, we felt compelled to represent it using some rigorous notation, namely UML class diagrams [5]. In addition to the final diagram, the construction process itself helps to better understand the domain. Also, we use *i** models for representing the relationships of among COTS domains (modeled as actors) in different contexts.

3. Quality Models in GESSI: Ongoing Work

Consider the organizational environment

Systems do not operate in isolation; they interact with external actors (human, software, organizations, etc.) which act together as its environment. External actors constrain the way in which the system behaves and determines the services it has to offer. In order to clearly understand the boundaries of the system, these actors

and their most important dependencies have to be identified.

We are currently incorporating this environment model as an additional artifact for quality models. One of the advantages of environment models is that can be used to identify the services that systems must offer. This is the base for the decomposition of a system-to-be into individual elements (system actors), which correspond to COTS domains, each offering well-defined services.

Add parameters to quality models

As part of the construction of quality models, metrics for quality factors can be proposed. But then a problem appears: some metrics cannot be precisely defined without taking into account some characteristics that are very particular of the system being considered. Therefore, we must abstract these particularities if we aim at reusing the quality model in several experiences. We are currently modeling these particularities by defining parameters that can be of two types, environmental and platform.

Environmental parameters often refer to the number of instances of a particular actor in the system. A typical case is the number of registered users in e-learning platforms, or the number of simultaneous connections supported by videoconference systems. Environmental parameters use to be fixed and with little margin of negotiation due to organizational constraints.

Some quality factors such as time efficiency, security or fault tolerance are greatly influenced by the underlying platform where the system is going to operate. Each platform strategy is based upon particular attributes which determine their influence on the system. These attributes are considered platform parameters. As an example, fault tolerance is greatly influenced by the layout of servers in the architecture. Servers may be organized into clusters, and clusters may be managed following diverse strategies (active-active, active-passive, etc.) that affect this subcharacteristic. Also, organization of discs concerning RAID level has a great impact on fault tolerance.

Combination of quality models

Individual quality models give an exhaustive but isolated view of quality. Often, we need to consider several COTS components to be combined to form a COTS-based system. Then, we need to build a composite quality model from the individual ones. One way to construct such composite models is by providing mechanisms to combine the elements of the quality models of its individual components. These mechanisms take the form of combination patterns of the elements of the individual models. The result is a quality model which includes elements of the quality models of its components, and also some new factors concerning architectural concepts that are relevant with respect to the combination itself.

Patterns for reuse

In addition to the taxonomy of COTS domains, reuse in our framework is supported by some types of patterns. We identify the following:

• Environmental patterns repository. Situations that appear over and over in different system environment models can be identified, refactored and defined as patterns. For each pattern, we state which requirements lead to it and which ISO/IEC 9126-1 subcharacteristics are addressed.

• Platform strategies repository. A catalogue of the most usual platform strategies used when deploying a system (such as clustering, load balancing, and so on) can be maintained. For each of them, the ISO/IEC subcharacteristics affected by their use are identified, as well as the platform parameters likely to appear.

• Quality patterns repository. When building individual quality models, there are several quality factors that appear over and over in virtually all COTS domains. For instance, the management of users and ACLs; the concept of throughput; the need to encrypt data; etc. We allow the definition of quality factor patterns able to be reused in the quality models for these COTS domains.

Our COSTUME for building composite quality models

Our former IQMC method for building quality models is being replaced by a more complete method that takes into account the concepts mentioned so far in this section. This new method, COSTUME (*CO*mposite *S*of*T*ware system q*U*ality *M*odel d*E*velopment), consists of 4 activities: 1) build an environmental model of the domain using *i** SD; 2) identify the COTS domains that play a part in the composite domain; 3) build individual quality models for them using IQMC; 4) combine the individual quality models using the defined combination patterns. As a result, we can build quality models useful for complex COTS-based systems procurement.

Connection with quality requirements

One of our aims is to use quality attributes in activities other than evaluation, for instance requirements elicitation and refinement. For this reason, we aim at bridging the gap between quality features and quality requirements. One way to do so is including the latter in the quality model that contains the former. Then, when starting a requirements elicitation and analysis process involving a quality model, the requirements therein form a baseline upon which the process can run.

Quality requirements can be bound to any node of the quality model hierarchy. More precisely, given a requirement that involves some quality features, it must be placed at the first common ancestor in the quality

model hierarchy. Of course, just those requirements that are not too system-specific appear in the quality model. Usually, they must be abstracted from a particular system-to-be to become reusable.

4. Conclusions

We have provided the main results and ongoing work of our research group in the field of COTS evaluation using quality models. Our results come from some industrial and also academic experiences in fields like ERP systems, mail server systems, requirements management tools, workflow technology and document management systems [5, 6, 7]. We aim at refining our models, processes and techniques with new ongoing and future experiences. As future work we plan to compare our approach with the NFR Framework and possibly to integrate their catalogue of types and correlation categories.

References

[1] ISO/IEC Standard 9126-1 Software Engineering – Product Quality – Part 1: Quality Model, June 2001.

[2] P. Botella, X. Burgués, J.P. Carvallo, X. Franch, G. Grau, J. Marco, C. Quer. "ISO/IEC 9126 in practice: what do we need to know?". In *Proceedings of First Software Measurement European Forum* (SMEF), 2004.

[3] X. Franch, J.P. Carvallo. "Using Quality Models in Software Component Selection". *IEEE Software*, 20(1), 2003.

[4] J.P.Carvallo, X. Franch, C. Quer, M. Torchiano, "Characterization of a Taxonomy for Business Applications and the Relationships between them". In *Proceedings of the 3rd International Conference on COTS-Based Software Systems* (ICCBSS), LNCS, 2004.

[5] J.P. Carvallo, X. Franch, C. Quer. "Defining a Quality Model for Mail Servers". In *Proceedings of the 2nd International Conference on COTS-Based Software Systems* (ICCBSS), LNCS, 2003.

[6] P. Botella, X. Burgués, J.P. Carvallo, X. Franch, J.A. Pastor, C. Quer. "Towards a Quality Model for ERP System Selection". Chapter of the book *Component-Based Software Quality: Methods and Techniques*, LNCS, 2003.

[7] J.P. Carvallo, X. Franch, C. Quer. "A Quality Model for Requirements Management Tools". To appear as a chapter of the book *Requirements Engineering for Sociotechnical Systems*, Idea Group, 2004.