ISO/IEC 9126 in practice: what do we need to know?

P. Botella, X. Burgués, J.P. Carvallo, X. Franch, G. Grau, J. Marco, C. Quer

Abstract

ISO/IEC 9126 is currently one of the most widespread quality standards. In its actual form it embraces both quality models and metrics. Due to its generic nature, some of the concepts presented therein need to be refined before using the standard in a real project. In our paper, we aim at exploring which are the concepts that require being more elaborated before putting the standard to work. Specifically, with respect to the part 1 of the standard we focus on the hierarchical form of the quality entities (i.e., characteristics, subcharacteristics and attributes) that appear in quality models; we propose some criteria to distinguish among subcharacteristics and attributes; we distinguish among derived attributes and basic attributes; and we propose an extension for covering not only technical factors but also managerial and political ones. Concerning the other 3 parts of the standard (one of them currently pending of approval), we distinguish different categorization criteria of metrics (scale, type, objectivity, qualitative/quantitative) and we make explicit the rationale behind their definition with the Goal-Question-Metric approach. As a final contribution, we show the use of UML class diagrams for representing all the concepts of the standard and their relationships, and we highlight the need for having tool support for quality model development and metrics definition.

1. Introduction

Software quality models [1] are artefacts used for describing the quality factors of a single software product of any nature (e.g., a bespoke application, a commercial off-the-shelf component or a web service) or a software domain (e.g., ERP systems or document management tools). A software quality model provides a taxonomy of *software quality factors*. Software quality models and their metrics may be used in many contexts, for instance during the development of a new application [2, 3] or when selecting commercial components [4].

There are many approaches to deal with the identification and representation of software quality factors. All of them rely on the existence of a catalogue of factors which in general is defined as extensible. There is also a great variety of those catalogues, many of them in the form of standards. In this paper, we focus on one of such standards, the ISO/IEC 9126 quality standard [5], which we have used in our experiences.

The ISO/IEC 9126 is currently one of the most widespread quality standards. In its actual form it embraces both quality models and metrics. Due to its generic nature, some of the concepts presented therein need to be refined before using the standard in a real software procurement project. In our paper, we identify those issues that require clarification before putting the standard to work, and we highlight also the role that conceptual models and tool support may play. Specifically, our paper addresses the following points:

- Some concepts appearing in the standard are not clearly defined. When we started to use the standard, this fact made harder return on investment and quality model reuse. Therefore we felt compelled at some time to define clearly each concept of the standard and the relationships among them.
- In addition to quality factors, there are other type of factors that are utterly important when procuring software systems. We have found that these factors may be managed in a

similar manner as quality ones, providing therefore a unified framework that has revealed to be comfortable during software procurement.

- The elements of a software metric were not made clear in the standard definition. For instance, it was not clear what type of metric scales were allowed, nor which classification the standard considers. We have decided then to make this information also explicit.
- The definition of the proposed metrics did not follow an stated procedure. However, their presentation seems to follow a layout similar to Basili's Goal-Question-Metric approach [6], which we think is appropriate. We have incorporated this method to our development process.
- The concepts enclosed in the standard, enriched with the decisions outlined in the precedent items, form together a rich base of knowledge. From our point of view, it becomes necessary to represent this knowledge in a more formal manner than in pure text, therefore we have developed a conceptual model using UML class diagrams [7].
- The development of large quality models with a rich catalogue of metrics, is far from being a simple task. We have defined a methodology for building ISO/IEC 9126-based quality models [8] and we have developed some tool support for facilitating this activity.

All of these points are addressed from section 3 to 8. Before that, we give an outline of the ISO/IEC 9216 software quality standard.

2. The ISO/IEC 9126 quality standard

The International Organization for Standardization (ISO) has defined a set of ISO and ISO/IEC standards related to software quality. First of all, it's worth to mention the ISO 9000-3 guidelines for applying the ISO 9001 standard concerned with quality assurance processes to the development, supply, installation and maintenance of computer software. Then, the standard ISO/IEC 9126 [5] for software product quality, which has to be used in conjunction with the ISO/IEC 14598 for the evaluation of software products. Other standards related to or that can be used in conjunction with ISO/IEC 9126 and ISO/IEC 14598, are:

- ISO/IEC 12119 - Quality requirements for software packages
- ISO/IEC 12207 - Software life cycle processes
- ISO/IEC 14143 - Software measurement
- ISO/IEC 15271 - Guide for ISO/IEC 12207
- ISO/IEC 15504 - Software process assessment (also known as *Spice*)
- ISO/IEC 15939 - Software measurement process

ISO/IEC 9126-1 specifically addresses the definition of quality models and thus it is chosen as our framework in this paper. The main idea behind this standard is the definition of a *quality model* and its use as a framework for software evaluation. An ISO/IEC 9126-1 quality model is defined by means of general *characteristics* of software, which are further refined into *subcharacteristics*, which in turn are decomposed into *attributes*, yielding to a multilevel hierarchy; intermediate hierarchies of subcharacteristics and attributes may appear. At the bottom of the hierarchy appear the measurable software attributes, whose values are computed by using some *metric*. Throughout the paper, we refer to characteristics, subcharacteristics and attributes as *quality factors. Quality requirements* may be defined as restrictions over the quality model. The 2001 version of the standard introduces the subcharacteristics as normative, based on the informative subcharacteristics introduced in the 1991 version.

The ISO/IEC 9126 standard makes a distinction between *internal quality* and *external quality*, and introduces the so-called *quality in use*. These models categorise software quality attributes into characteristics. The attributes that can be measured during the development

process are referred to as internal. The external behaviour can be measured during the testing process, and finally the user's view of quality is shown measuring the quality-in-use attributes. Examples of the metrics for external, internal and quality-in-use attributes can be found respectively in parts 2, 3 and 4 of the standard (this last part not yet published).

The quality model introduced in the standard is common for external and internal quality aspects (although of course they will be refined in different ways), whilst another model for quality in use is introduced. Table 1 enumerates the six quality characteristics defined in the ISO/IEC 9126-1 internal/external quality model and their decomposition into subcharacteristics. This quality model is chosen as framework in this paper. Just to mention, the quality model for quality in use is categorised into four characteristics: effectiveness, productivity, safety and satisfaction.

Table 1: the ISO/IEC 9126-1 internal/external quality model

Characteristics	Subcharacteristcs
Functionality	suitability
	accuracy
	interoperability
	security
	functionality compliance
Reliability	maturity
	fault tolerance
	recoverability
	reliability compliance
Usability	understandability
	learnability
	operability
	attractiveness
	usability compliance
Efficiency	time behaviour
	resource utilisation
	efficiency compliance
Maintainability	analysability
	changeability
	stability
	testability
	maintainability compliance
Portability	adaptability
	installability
	co-existence
	replaceability
	portability compliance

3. Refinement of ambiguous concepts in the ISO/IEC 9126-1

The concept of quality model is defined in different ways by different organizations and authors. Furthermore, even a particular definition may contain some ambiguities and (sometimes intended) incompleteness that become apparent once quality models must be built. We think that return on investment can be enhanced by fixing these ambiguities and incompleteness up to a maximum extent without compromising the generality of the chosen approach.

This is the case in the ISO/IEC 9126-1. As mentioned in the previous section, the standard states three approaches to quality (internal quality, external quality and quality in use). Internal and external quality share a set of six high-level factors or characteristics (functionality, reliability, usability, efficiency, maintainability and portability), each of them divided in a few subcharacteristics. Quality in use is determined by a set of four characteristics (effectiveness, productivity, safety and satisfaction). The standard also defines as final target the definition of the lowest-level quality factors or *attributes* together with the metrics to use in measurement. However, the standard is not precise in none of the following points:

- *Are hierarchies of subcharacteristics and attributes allowed?* In our refinement we have decided to allow it, for not restricting in this aspect the hierarchies. On the one hand, subcharacteristics may be decomposed into other subcharacteristics or alternatively into attributes. On the other hand, attributes may be derived and basic. Derived attributes may be decomposed into other attributes, where basic attributes may not.
- *Could subcharacteristics and attributes be related with more than one quality factor in its upper level of the hierarchy? Or they should form a perfect hierarchy?* In our refinement we have decided to restrict the hierarchy of characteristics and subcharacteristics in a perfect hierarchy form. However, we allow overlapping in the hierarchy of attributes and their relationships with subcharacteristics. We deal in a different way for attributes since specific attributes may contribute to more than one subcharacteristic. We have observed in our experiences that it is not the case of subcharacteristics and characteristics, as a consequence of their quality factors classification role.
- *Should characteristics and subcharacteristics have metrics defined for their measurement?* In our refinement, characteristics are non-measurable factors and they are just used as a way of classifying the upper level of subcharacteristics. On the contrary we allow subcharacteristics to be measured if needed. Although in this case it is not possible to assign them *objective metrics* (i.e., formal and precise ways to measure them in terms of the values given to the subcharacteristics or attributes which they are decomposed into), we think that we can give them *subjective metrics* (i.e., ways to measure them depending on the perception of the people involved in the measurement process).
- *In case of having hierarchies of attributes, are all attributes measured in the same way?* In our refinement we have derived and basic attributes. All of them are measurable quality factors. Specifically, metrics for basic attributes must be objective, since they should be measured in a formal and precise way independent of the thought and perception of the people involved in its measurement. For derived attributes sometimes it is not possible to find an objective metrics to measure them in terms of the values of attributes in which it is decomposed. In these cases a subjective metrics is required.

The definitions of the different types of quality factors deduced from the above statements are included to end this section:

- *Characteristics* in a quality model are non-measurable quality factors used with the aim of classification of the upper level of subcharacteristics of the model.
- *Subcharacteristics* in a quality model are quality factors that may be subjectively measured when required, and may be decomposed into other subcharacteristics or alternatively into attributes that help in its measurement. A characteristic and its subcharacteristics result in a perfect hierarchy.
- *Attributes* can be derived or basic. Attributes may be related with more than one quality factor in its upper level of the hierarchy.
- *Basic attributes* are objectively measurable quality factors that are not further decomposed.

- *Derived attributes* are measurable quality factors that may be decomposed in one or more attributes. Sometimes it is not possible to find an objective metric to derive its value in terms of the attributes in which it is decomposed. In these cases a subjective metric is required.

4. Consideration of non-technical factors

The ISO/IEC 9126-1 standard has been conceived as a basis for the construction of quality models oriented to the evaluation of the technical factors (functional and non-functional) of a quality scope. Non-technical factors (e.g. managerial, economical or political) have not been considered by the standard. Despite this fact, in many situations these factors are also significant, often even more important than technical ones; therefore, not considering them could compromise the success of the undertaken activity.

A convenient and coherent way to handle non-technical factors consists in structuring them in the same way as the technical ones in the ISO/IEC standard. That is, we have defined a set of high level characteristics and subcharacteristics, which may be detailed in the usual way by spanning hierarchies of non-technical subcharacteristics and attributes. As a result, we use a enlarged quality model including both types of factors and therefore it becomes more applicable. Table 2 shows an excerpt of the non-technical factors structure that we use.

Table 2: a categorization of non-technical factors following the ISO/IEC 9126-1 style (fragment)

Characteristic	Subcharacteristic	Attribute
Vendor	Economy	Market share
		R+D budget
	Reputation	Years in the market
		Certification of the process
		Directory of clients
	Support	Distribution channel
		Offered services
		Location
Product	Distribution	Commercialisation strategy
		Licence cost
	Stability	Time in the market
		History of versions

5. Definition of the metrics

We find the use of the terms “internal” and “external” in the standard is somewhat confusing. On the one hand, they are applied to “quality” to indicate the view of a product isolated from its environment or the visible effects of the product. On the other hand, they are applied to metrics sometimes to indicate which quality attributes (internal or external) are being measured and sometimes depending on how are metrics computed (observing the product or its effects). We take the option that classifies metrics as internal or external depending on how they are computed. We think that the none of the four possible combinations between internal and external quality factors and metrics is to be rejected, although some will appear more frequently than others.

We feel that the role of metrics in the 2001 version of the standard is beyond the convenient one. The 1991 version of the standard advocated the definition of a quality model as a hierarchy from a fixed set of characteristics to the attributes. It was supposed that metrics

would be defined to measure the quality factors of the model. Now, metrics are embedded in the model itself attaching one or more for each subcharacteristic in the model. We prefer to proceed as it was suggested by the past version; anyway, it is possible to obtain a result conforming the 2001 version.

6. Use of the Goal-Question-Metric approach

There are many proposals for identifying and defining metrics. We have incorporated in our proposal one of the most widespread approaches, the Goal-Question-Metric (GQM) [6]. In GQM, goals of the product under measurement are identified, and then some questions are raised to characterize the way the assessment of a specific goal is going to be performed. Last, a set of metrics is associated with every question in order to answer it in a quantitative way; metrics can be objective or subjective. The final result of the GQM approach is a hierarchical structure in graph-like form, since metrics may influence in more than one question, and questions may be related to more than one goal.

GQM goals are composed of four elements: purpose, issue, object and viewpoint. The GQM approach recommends to identify quality goals for issues; therefore, we tailored the GQM defining one issue for each attribute of the ISO/IEC-based quality model of the product under measurement. As a consequence, we have as many goals as quality attributes.

As an application example, we show how we have applied the GQM to asses the quality of a proposal for the prospective Ada Standard Container Library [9] under an ISO/IEC-based quality model defined for this library [10]. Table 3 shows the attributes (name, definition and examples) that play a part on the *Core Suitability* subcharacteristic, belonging to the ISO/IEC 9126-1 *Suitability* subcharacteristic, defined as the types of containers offered by the library and their implementations. These types are mainly characterised by the operations for adding, removing, modifying and searching elements.

Table 3: quality attributes for the Core Suitability subcharacteristic

Core Suitability		
Attribute	Definition	Examples
Category variety	Range of different categories of containers offered by the library	Sequences, maps, sets, trees, graphs
Container variety	Range of different containers provided by every category	For sequences: stacks, queues, lists
Implementation variety	Range of different implementations provided by every category	For maps: closed hashing, red-black trees
Operation variety	Range of different operations provided by every container	For stacks: empty, push, pop, top, isEmpty

Table 4 presents a summary of goals, questions and metrics for the core suitability attributes. Questions are the same for the four goals bound to the four core suitability attributes: expressive power embraced by the library, and quality of its contents. With respect to expressive power, we felt necessary to distinguish among basic and advanced categories, types of containers, implementation strategies and operation sets. The concept of basic depends on the viewpoint (this is marked in the table with the italics style). So, the Ada community may classify e.g. graphs as a basic category of containers or as advanced one, depending on their particular requirements. In this example we focus on basic elements, to simplify metrics definition; a deeper analysis shall consider advanced elements yielding to additional metrics. Therefore, the metrics are defined as a measure of the basic elements of each kind present in the library.

Table 4: GQM model for the Core Suitability quality attributes

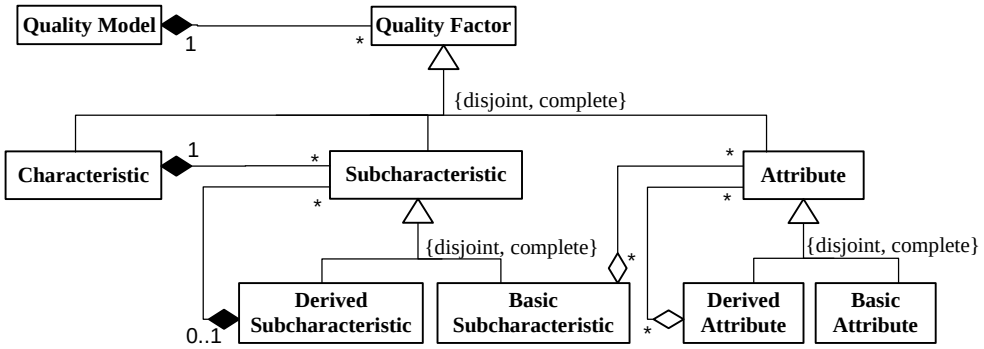
Goal	Question	Metric
Purpose: Have an appropriate Issue: Category variety Viewpoint: Ada community	Has the library the most frequently used categories of containers?	% of *basic* categories
	Has the library a robust proposal of categories of containers?	100 - % of unnecessary categories
Purpose: Have an appropriate Issue: Container variety Viewpoint: Ada community	Has the library the most frequently used types of containers for each category?	mean(% of *basic* types of containers for each category it has)
	Has the library a robust proposal of types of containers?	100 - % of conflicts among two *basic* types of containers
		100 - mean(% of *unnecessary* types of containers in each category it has)
Purpose: Have an appropriate Issue: Implementation variety Viewpoint: Ada community	Has the library the most frequently used types of implementation strategies for each category?	mean(% of *basic* implementation strategies for each type of container it has)
	Has the library a robust proposal of implementation strategies for each container?	100 - mean(% of *unnecessary* implementations of containers in each category it has)
Purpose: Have an appropriate Issue: Operation variety Viewpoint: Ada community	Has the library the most frequently used operations for each container?	mean(% of *basic* operations in each type of container it has)
	Has the library a robust proposal of operations for each container?	100 - mean(% of *unnecessary* operations in each type of container it has)

7. Conceptual model

In order to allow a precise understanding of the standard ISO/IEC 9126 and the refinement and enrichment proposals presented in this paper, we have developed a conceptual model using UML class diagrams. On the one hand, this model gives an unambiguous and complete view of the quality framework and allows a precise understanding of all the enclosed concepts and relationships between them. On the other hand, it facilitates the construction of quality models, since quality models will be object models of this conceptual model. Finally, it can be used as the basis for the development of a tool for supporting the quality model construction. We have chosen UML [7] as language because in the last years has become a standard *de facto*.

In figure 1 we include an excerpt of the UML class diagram that describes the hierarchical structure of the quality models for the enriched standard. Every class in the diagram states a concept in the quality framework described. Relationships allow giving a precise view of how quality models are structured in quality factors. Note that we have specialized the class subcharacteristic into two subclasses, in order to avoid a specific subcharacteristic to be decomposed into both subcharacteristics and attributes at the same time. Therefore, a subcharacteristic is either decomposed into other subcharacteristics or into attributes. Although in a quality model, quality factors will be at the end always decomposed until the level of basic attributes, we have decided to reflect in the multiplicity of the relationships the possible states in which the quality model is not completely constructed. Thus, during this construction is admissible to have a characteristic, a subcharacteristic or a derived attribute still not decomposed.

Figure 1. UML conceptual model for the enriched ISO/IEC 9126 standard (OCL restrictions are not included)



8. Tool Support

The refinement of the standard that we propose aids in constructing quality models based on the ISO/IEC 9126 standard, but it also adds complexity to the original definitions. Due to this, the use of a tool that supports all the concepts becomes a crucial key point.

In figure 2 we show the layout of the **QM** tool that our group has developed in-house with the aim of providing support to the construction of quality models following the ideas of this article. The new quality models can be created using the ISO/IEC 9126-1 standard as a departing point. The quality entities are viewed in their hierarchical form and, when creating a new quality entity, the restrictions exposed in section 3 are applied: characteristics can only be decomposed into subcharacteristics, subcharacteristics into subcharacteristics or attributes and derived attributes can be decomposed into attributes until a basic attribute level is defined. Basic attributes cannot be further decomposed. The tool also allows to modify the departure quality model with non-technical factors as proposed in section 4. The definition of metrics for evaluating the quality factors can be done in the way explained in section 5: characteristics are considered non-measurable, but it is possible to define subjective metrics for subcharacteristics and basic attributes and direct measures for basic attributes. Derived attributes can also be evaluated in terms of the basic attributes that decompose them.

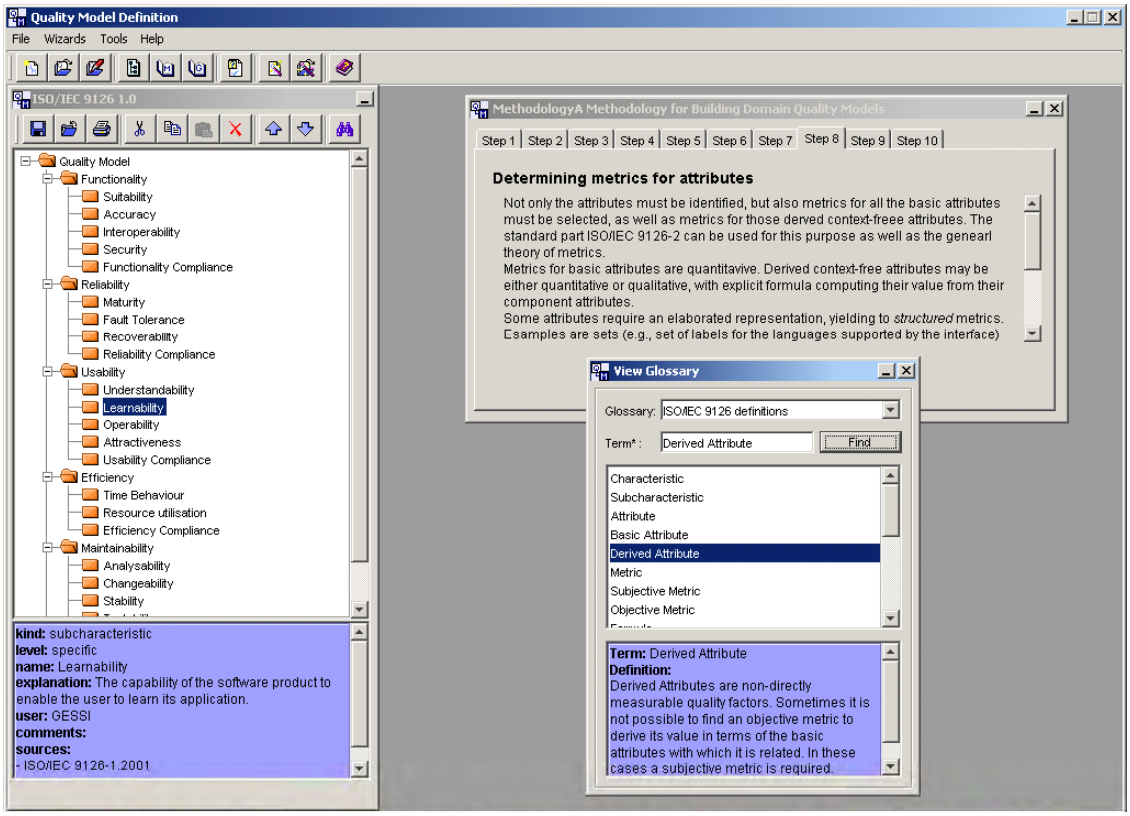
The tool has been designed using the conceptual model showed in section 7 and also provides, among others, the following features: the definition of requirement patterns, the establishment of relationships between quality factors, the reuse of pieces of quality models, the definition and use of methodologies for constructing the quality models, and a glossary of quality terms.

The current version of the **QM** tool follows a client/server architecture based on the JAVA™ *servlets*, communication among subsystems is implemented using HTTP/XML and the data is stored in a MySQL database.

9. Conclusions

In this paper we have commented some practical issues arising when using the ISO/IEC 9126 quality standard. Our knowledge comes from nearly a dozen of both academic and industrial experiences, mainly in the field of COTS selection. Some of this experiences are partially reported in [10, 11, 12].

Figure 2: a snapshot of the QM tool for building quality models



The main contributions of our work with this standard can be summarised as follows:

- The development of a conceptual model that defines in a precise way all the concepts that appear in the standard, both belonging to the part 1 (quality model) and the other parts (metrics).
- The definition of a methodology for extending the ISO/IEC 9126-1 quality model.
- The extension of the standard with some non-technical factors to leverage all kind of relevant criteria.
- The use of the Goal-Question-Metric approach to provide a rationale to the definition of software metrics.
- The construction of tool support to make easier the development and maintenance of quality models and metrics.

10. Acknowledgments

This work has been partially supported by the Spanish project TIC2001-2165.

11. References

[1] International Organization for Standarization, "ISO Standard 8402: Quality management and quality assurance - Vocabulary", International Organization for Standarization, Geneve, 1994.

[2] Dromey, R.G., “Cornering the Chimera”, IEEE Software, 13(1), 1996, pp. 33-43.

[3] Kitchenham, B., Pfleeger, S.L., “Software Quality: the Elusive Target”, IEEE Software, 13(1), 1996, pp. 12-21.

[4] Franch, X., Carvallo, J.P., “A Quality-Model-Based Approach for Describing and Evaluating Software Packages”, in 10th IEEE International Conference on Requirements Engineering (RE 2002) conference proceedings, Essen (Germany), September 2002.

[5] International Organization for Standarization, "ISO Standard 9126: Software Engineering – Product Quality, parts 1, 2 and 3", International Organization for Standarization, Geneve, 2001 (part 1), 2003 (parts 2 and 3).

[6] Basili, V.R., Caldiera, G., Rombach, H.D., “Goal Question Metric Paradigm”, in J.J. Marciniak (ed.), Encyclopedia of Software Engineering 1, John Wiley & Sons, New York, 1994, pp. 528-532.

[7] Booch, G., Rumbaugh, J., Jacobson, I., “The Unified Modeling Language User Guide”, Addison-Wesley, Reading, 1999.

[8] Franch, X., Carvallo, J.P., “Using Quality Models in Software Package Selection”, IEEE Software, 20(1), 1996, pp. 34-41.

[9] Marco, J., Franch, X., “A Framework for Designing and Implementing the Ada Standard Container Library”, in 2003 ACM SIGAda Annual International Conference proceedings, ACM Press, San Diego (CA, USA), December 2003.

[10] Franch, X., Marco, J., “A Quality Model for the Ada Standard Container Library”, in 8th Ada-Europe International Conference on Reliable Software Technologies conference proceedings, Springer-Verlag, LNCS 2693, Toulouse (France), June 2004.

[11] Carvallo, J.P., Franch, X., Quer, C. “Defining a Quality Model for Mail Servers”, in 2nd International Conference on COTS-Based Software Systems (ICCBSS) conference proceedings, Springer-Verlag, LNCS 2580, Ottawa (Canada), February 2003.

[12] Botella, P., Burgués, X., Carvallo, J.P., Franch, X., Pastor, J.A., Quer, C., “Towards a Quality Model for the Selection of ERP Systems”, in Cechich, A., Piattini, M., Vallecillo, A. (eds.), Component-Based Software Quality: Methods and Techniques, Springer-Verlag, LNCS 2693, 2003, pp. 225-246.