ISSN:0975-9646

General Meta Model of Software Quality

Adil KHAMMAL[*1], Youness BOUKOUCHI[#2], Mohamed Amine HANINE[+3], Abdelaziz MARZAK[*4]

** Departement of Computer Sciences, Faculty of Sciences Ben M'Sik, Hassan II University*
 # Departement of Computer Sciences, National School of Applied Sciences, IbnZohrUniversity
+ Departement of Computer Sciences, Sciences and technologies Faculty, Hassan I University

ABSTRACT: **According to their purpose, most quality models incorporate a variety of different information in a quality model. However, for the best management of software quality throughout the entire life cycle of a software product, we need to combine these isolated models to achieve a more complete picture of the quality of software. To overcome the ambiguity and incompleteness of the software quality models, we must define a formal quality Meta model. In this paper we propose a three-step methodology for the construction of a general Meta model of software quality. This methodology is based essentially on the principles of Model Driven Architecture and factoring of concepts.**

***Keywords*—Quality Meta Model, Software Quality, Quality Model, Model Driven Architecture.**

1. INTRODUCTION

The software quality is more and more seen as an influential critical parameter in business, it's an important motive for customer satisfaction. Software quality is a fairly complex and multifaceted concept, In order to get a complete picture of the software's quality, we use a quality model (McCall, ISO9126...). In [16] we proposed a reference framework to characterize and compare the software quality model.

This framework considers four perspectives, known worlds of the subject, of the usage, of the system and of the development of software quality. These four worlds are explained by facets and their values. This framework has allowed us to characterize and compare a number of software quality models, and also allowed us to establish several observations : Lack of decomposition criterion in hierarchical models Quality attributes redundancy; Some models do not cover the entire life cycle of a software; Most models studied do not have a clear vision to explain the correlation between metrics and criteria, such as when a criteria gets a low score, it is difficult to link the score to directly point out the issue, especially when the criterion is made up of several metrics; Also most of these models suffer from the absence of guidelines and criteria for decomposition of complex quality concepts, making it difficult for them to be sophisticated and even located in some large quality models.

In other words, this reference framework has allowed us to highlight the need for a Meta model, the aim of which is not only operationalizing the existing software quality models but also correcting their general oversights and limitations.

In this paper we propose a three-step methodology for the construction of a general Meta model of software quality. This methodology is based essentially on the principles of Model Driven Architecture and factoring of concepts. We present initially the concepts of quality, then we briefly present our reference framework for software quality, then we present our methodology for building the overall Meta model and finally we apply this methodology.

2. MODEL AND META MODEL OF SOFTWARE QUALITY

A. Model of software quality

The software quality is more and more seen as an influential critical parameter in business, it's an important motive for customer satisfaction. Any absence of software quality can cause heavy financial losses, a dissatisfaction of the users, and the damage to the environment which can even result in deaths as ultimate and grave consequence. To obtain a complete image of the quality of a software we call on the models of quality which contain rules describing what must be a software of quality, they area well accepted way to define, assess and predict software quality.

Examples of such hierarchical models were used first by Boehm [1] and McCall [4] and later adapted by ISO 9126 [3]. These models define quality by decomposing with well-known quality criteria such as functionality, reliability, usability, efficiency, maintainability and portability which in turn are subdivided into more specific sub-criteria.

Major contribution of McCall's model [4] is consideration of relationships between quality characteristics and metrics. Quality is classified into revision, operation and transition perspective, considering user's and developer's view. Boehm [1] introduced quality model to automatically and quantitatively evaluate the quality of software. In this model, characteristics and sub-characteristics are loosely-coupled and it's (as-is) aspect is subjectively specified. FURPS [14] model consider two steps, setting priorities and defining quality attributes that can be measured. Dromey [2] taken into consideration relationship between characteristics and sub-characteristics in its product based quality evaluation framework and emphasized that to make a high quality product all constituent artifacts must be of high quality, so he made a product based quality model, but he failed to discuss how it could be realized. ISO 9126 [3] quality model is based on McCall and Boehm's model which cover all aspects of software quality but metrics are not consistent with their own definitional concept of metric.

These hierarchical models have been the basis of several adapted models for specific needs or projects. Their objective is to capture the knowledge on quality present in hierarchical quality models, guidelines, and measurement tools in one comprehensive quality model.

In [10] Bertoa proposed a quality model containing a set of quality attributes and measurement of these attributes for effective evaluation of COTS components. GEQUAMO is a generic model of software quality proposed by Geqrgiadou [11]. Ortega [12] defined a systemic approach to software products in proposed quality model. This model is evaluated using a method so it can be validated and also enhanced. The software quality model proposed by Andreu [13] is based on ISO 9126 which may be used for development and evaluation of original components and may be tailored according to the organization re-user and the domain needs of the targeted component. Behkamal [6] also proposed a model based on ISO 9126 which may be used for evaluation of B2B applications. In this model, quality factors are extracted from web applications and B2B e-commerce applications. In [5] Sharma added/modified some extra features to ISO 9126 to make it appropriate for given applications and for weight assignment Analytical Hierarchical Process (AHP) is used. The result value of AHP can be used to compare and select the best suitable component as per all desired quality characteristics. Srivastava's [7] model measures the software quality statistically by taking care of three different views of user, developer and manager. Srivastava [7] proposed a model of software quality that takes into account the three different views of the user, developer and manager. Kumar [9] proposed an aspect oriented software quality model (AOSQUAMO), in this AHP is used to evaluate quality of AO software systems as a single parameter. Carvahlo's [8] proposed quality verification framework may be used to evaluate the quality of embedded software components.

B. A reference framework for quality software

Software quality is a fairly complex and multifaceted concept, In [16], we proposed a reference framework (Figure 1) to characterize the software quality models. This framework considers four perspectives, known worlds of the subject, of the usage, of the system and of the development of software quality. These four perspectives are explained by facets and their values. We have applied this framework to quality models. This application has revealed that none of the models covers all facets of the framework, especially the system and development facets, which explains the gap between these models and their operational use. This piece of work has allowed us to highlight the need for a Meta model, the aim of which is not only operationalizing the existing software quality models but also correcting their general oversights and limitations.

We have applied this reference framework to quality models. The software quality models that we studied are not complete as our frame of reference. Indeed, various facets are not covered by some models. Also, no model covers all frame facets. This means that the models are not complete but fragmented. Gaps are not obvious to the use and subject worlds. Nevertheless here are some of the issues faced by the quality models in these two worlds: Absence of an explicit model; Lack of decomposition criterion in hierarchical models; Quality attributes redundancy (one inside the other such as safety which is strongly influenced by the availability being a part of reliability); Some models do not clearly tell apart the different perspectives of their use; These models are usually limited to a fixed number of levels, which limits the definition and structuring of complex quality attributes into three or four levels, making the decomposition of some factors to measurable properties challenging; Some models do not cover the entire life cycle of software.

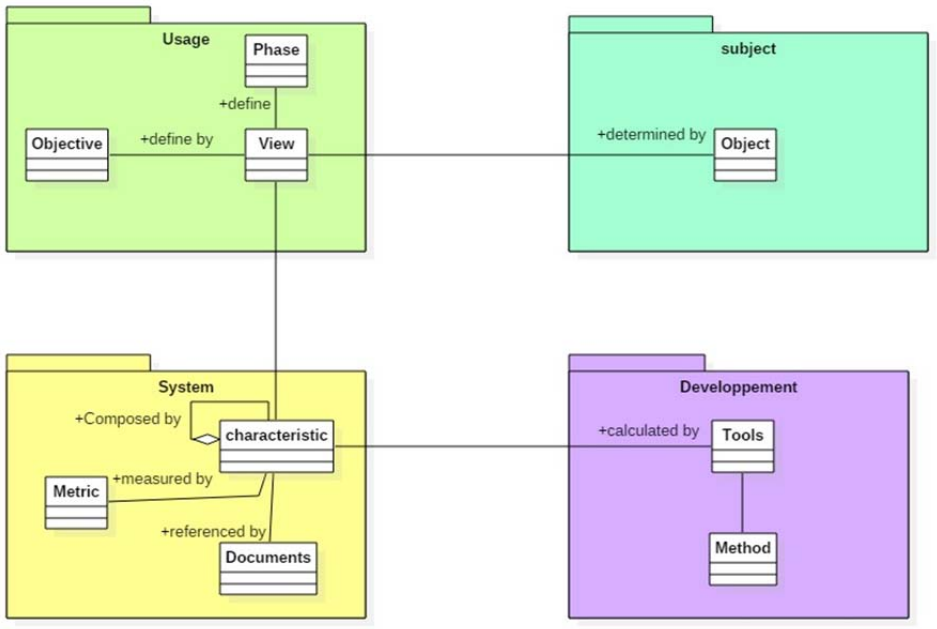

Figure 1: Reference framework

As far as the system and development world are concerned, which respectively own disclosures on the subject world and the tools to achieve objectives of software quality, shortages are particularly egregious for software quality models. This explains the gap between these models and their operational application. The problems of quality models in these two worlds are mainly due to the "Tools" and "Metric" facets, which we sum up as follows: Most models studied do not have a clear vision to explain the correlation between metrics and criteria, such as when a criteria gets a low score, it is difficult to link the score to directly point out the issue, especially when the criterion is made up of several metrics; Also most of these models suffer from the absence of guidelines and criteria for decomposition of complex quality concepts, making it difficult for them to be sophisticated and even located in some large quality models; Some models are not that simple tobe implemented in an environment because of the amount of defined criteria and metrics; Due to the lack of clear semantics, the aggregation of measured values remains complicated; Models are not included in all the various tasks related to quality; There is no clear definition of the way in which we use a model.

In other words, this reference framework has allowed us order to characterize and compare different models of software quality on one hand. On other hand, we aim to highlight the key elements that must be considered to provide a Meta model of software quality.

C. Meta model of software quality

Most quality models mentioned above include a wide variety of different information in a quality model. Therefore, their main challenge is to find an adequate mechanism to structure the information. The structuring mechanism must allow unambiguous definition, without duplication, without contradiction of terms and concepts. In addition, the mechanism should also be able to relate the abstract quality characteristics with specific properties, components and measures to quantify. As we have seen in [16] These objectives are generally not respected by the quality models. The majority of these models are very useful in various fields. However, for proper management of software quality throughout the entire life cycle of a software product, we need to combine these isolated models to achieve a more complete picture of the quality of software. To overcome the ambiguity and completeness of non-quality models problems, we must then define a formal quality Meta model.

Independent of the modelling technique used to build a quality model, Deissenboeck [15] considers the existence of a defined Meta model as crucial. Even though, in many cases, quality Meta models are not explicitly defined for existing quality models. Accordingly, he defines: **Quality Meta Model** - A model of the constructs and rules needed to build specific quality models.

A quality Meta model must be generally easy to use, cover all aspects of quality, consider all possible characteristics of the underlying areas, be flexible enough to be applicable in all areas of application with modifications or minor additions. It must meet all the needs of interested-parties and must serve as a standard for evaluation of software products. So to do this, it must appeal to processes and tools of metamodeling that will allow a quick understanding of the concepts defined in different models and will also facilitate the handling of these models and concepts using a Meta model.

3. DESCRIPTION OF OUR METAMODELING APPROACH

In this section, we explain the methodology adopted for the development of the Meta model of software quality. Based on the model driven and factoring concepts engineering, we defined a three-step approach. This approach allows to get close and create a single integrated model, a Meta model that we call General (PIM GPIM), starting from several different models or standards.

As we mentioned, a software quality meta-model must cover all aspects related to the field of quality, it must also contain all possible characteristics of the underlying areas. Our meta-model, therefore, will consist of the concepts of the different models of quality and also concepts from measurement models as this is a very important field of software quality which is focusing on metrology-related concepts software.

Step 1: Construction of PIM: Faced with a multitude of standards and models using different names often for the same concept of software quality, the objective of this step is to capture the key concepts of the Meta model. A concept can be defined as the intellectual representation of an abstract idea. This is the idea that one has on a thing by detaching it from its real object.

In order to capture different concepts, we have to proceed first to a transformation of PSM of the models chosen to build the Meta model into a PIM whichconsists of a number of concepts (classes). At the end of this stage we will have a list of concepts used in different models.

Step 2: Factoring of concepts: The objective of this step is to compare the extracted different PIM concepts in order to factor them in a PIM General. Once the concepts are selected, we will analyze the differences and reconciliations. To do this, we perform a mapping between the concepts of these models. This mapping should determine:

- The levels of abstractions of selected models;
- For each concept model, its relationship to the concepts of the other model;
- A correlation level to qualify each relationship.

To make this comparison, we will define the terms used in the process that leads to the factorization of concepts.

- **A specific concept** is a concept belonging to a single software quality model.
- **A common concept** is a concept existing in at least two sources and models with a minimum set of common characteristics.

- **A reference concept** is a concept that belongs to our frame of reference.

The general Meta model of software quality (General GPIM) consists of reference concepts identified in our reference framewok and maximum intersection of all common concepts among all quality models sources.

Each model will be for us a PIM that contains several concepts of a software quality model, so we can write:

$$\mathbf{PIM_i = \{ C_j , \forall j \in N, C_j \in PIM_i\}}$$

$$\mathbf{GPIM = PIM_{ref} \cup \{ \cup_i^j PIMi \cap PIMj \quad / \quad i \neq j \}}$$

Step 3: Building a GPIM:

In this step we will proceed to: Solve all contradiction relations; Avoid duplication of concepts by consolidating them to the relationships with identity and inclusion; Maximize the potential for synergies by combining complementary concepts. Not taken into account concepts can be of two types:

- no correspondence exists in the mapping done;
- one (or more) connection (s) exist(s) but with low or average levels of correlation.

4. General Meta Model Of Software Quality

A. *Construction des PIM*

1) *Hierarchical models:* In order to capture different concepts, we begin with a transformation of PSM hierarchical models to a PIM which consists of a number of concepts (classes). We defined the specific structure of each quality PSM model, to move to a higher level of abstraction that includes all the common concepts of hierarchical models. Generally, these models are divided into hierarchical elements with the following structure: factors, sub-factors, criteria and metrics. The following table (Table 1) presents a comparison of the structure of these models.

2) From the comparison above between quality models (Table.1), we proposed [17] PIM (Figure 2) for all existing hierarchical models, it can generate models as ISO9126, MacCall,... or generate personal models according to the requirements of the designer (User, developer, etc.).

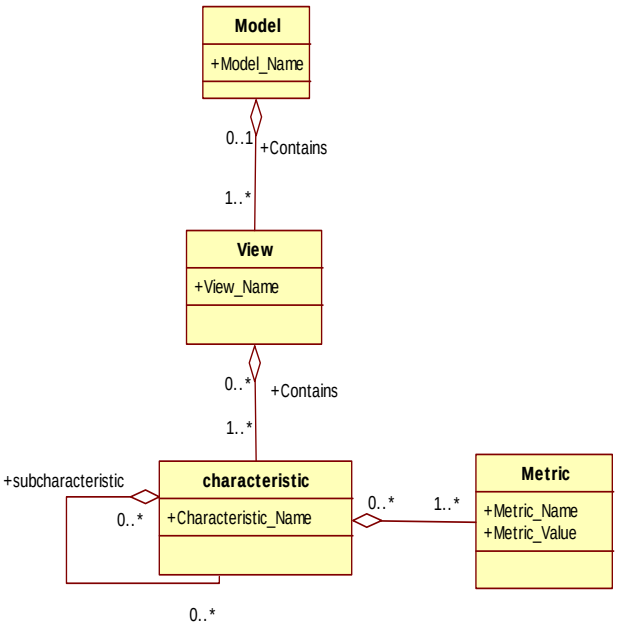

Figure 2: PIM of hierarchical software quality models

This PIM model (Figure 2) is divided into hierarchical elements. It structure quality is divided into three levels: view, characteristic and metric, whose characteristics can be divided into several subcharacteristics and so on.

1. Overview (Point of view): Quality can be perceived with various points of view, differences of views are mainly due to the fact that the project has many stakeholders, each stakeholder perceives the quality of its manner, what implies a prospect focused on the specific requirements of stakeholder towards the system.
2. Characteristic: After the view, we find the characteristics, (called Factors, Goals, Properties, etc), these characteristics are broken up into several under-characteristics until arrived in granular indecomposable characteristics and which are directly measurable by metrics.
3. Metric: A metric used to measure and evaluate a characteristic by values.

Table 1: A Comparison Between The Structures Of Model Quality Software

	Mac Call	Bohem	ISO	GQM	IEEE 1061	Dromey
Level 1	View	View	View	View	View	View
Level 2	Factors	high-level characteristics	Characteristics	Goals	Factor	Product properties
Level 3	Criteria	Intermediate level characteristics	Sub-characteristics	Questions	Subfactor	Quality attributes
Level 4	-	primitive characteristics	Quality Attributes	-	-	sub-attributes
Level 5	Metric	Metric	Metric	Metric	Metric	Metric

3) *Non-hierarchical models:* Regarding the non-hierarchical models (Quacomo, QUIMERA ....), the major part of them are already presented in the literature as PIM class diagram. We will use the PIM models proposed in [20] [21] [22] [23]

4) *Model of software Measurement:* In order to appropriate concepts dedicated to the specification of measurement and estimation methods we will base mainly on proposals of F. Ruiz [18] and F. Garcia [19].

After this first stage we will have a list of concepts used in different models.

B. Factoring of concepts

In this step we will list the concepts of the different models that we use to build our general Meta model of software quality GPIM. This list will allow us to categorize these concepts according to their type: Concept of reference, common concepts or specific concepts (Table 2).

After analyzing the various PIM, we could identify potentially factorable concepts in the general Meta model GPIM. So we have a list of concepts that must be analyzed and refined to build the general Meta model of software quality GPIM.

Table 2: A Comparison Between The Concepts

Model	Concepts		
	Reference concept	**Common concept**	**Specific concept**
PIM of refence framework[16]	Objetc, Phase, View, Objective,charateristic, Metric, Documents, tools, Methode	-	-
PIM of hierarchical models[17]	View, Characteristic, Metric	-	-
Quacomo[20]	Factor (Characteristic), Proprty (SubCharacteristic), EntityType (Objet), Measure (Metric),	QualityRequirement	Impact, QualityAspect, ImpactEvluation, QAspectEvluation
QUIMERA PIM [21]	Criterien (Characteristic), Artifact (Objet), Metric	Attribute, EvaluationMethod, Practice, Recommandation,	Limits
SQMREA PIM [22]	Characteristic, Metric, Artifact (Objet)	Attribute, Need, Requirement	Quality, SoftwareProduct, QualityLevel
PIM Parastoo et al. [23]	QualityGoal (Objective), ViewPoint(View),	EvluationMethod, Practice, Purpose,	QualityFramework, QualityCarryingProporty, Target,
PIM F. Ruiz et al [18]	Artefact, Defined Metric, Information Needs, Information Product, Measurable Object, Measurable Concept, , Measurement Method, Measurable Method,Metric, Analysis Model, Measurement Function,	Decision Criteria, Indicator, Base Metric, Derived Metric Attribute,	Activity ,Agent, Element, Interpretation, Measurement, Observation
PIM F. Garcia et al [19]	Measurable Concept,Entity, Entity Class, Information Need, Measurement Approach, Measurement Method, Measurement Function, Analysis Model, Measure,Quality Model	Decision Criteria, Indicator,Base Measure, Derived Measure,Attribute	Scale Type of Scale Unit of Measurement Measurement Measurement Result

C. Meta-model building:

In order to build our General Meta model of software quality, we must resolve all contradictions of relationships between concepts, avoid duplication of concepts by consolidating relationships with identity and inclusion. In other words, some of the concepts in the list are likely to be in GPIM while others must be specifically analyzed, validated and renamed.

We produce candidate field's concepts for GPIM (Figure 3). These concepts are concepts observed in both the PIM of our frame of reference and in different PIM of other models. These concepts can also be present in both abstractions in both PIM.

The concepts of our meta-model are divided into four separate packages related to each other:

- Package Subject: contains Artefact oriented concepts
- Package Usage contains the concepts oriented objective model
- Package System: contains Quality oriented concepts
- Package Development: contains oriented measurement and tools concepts

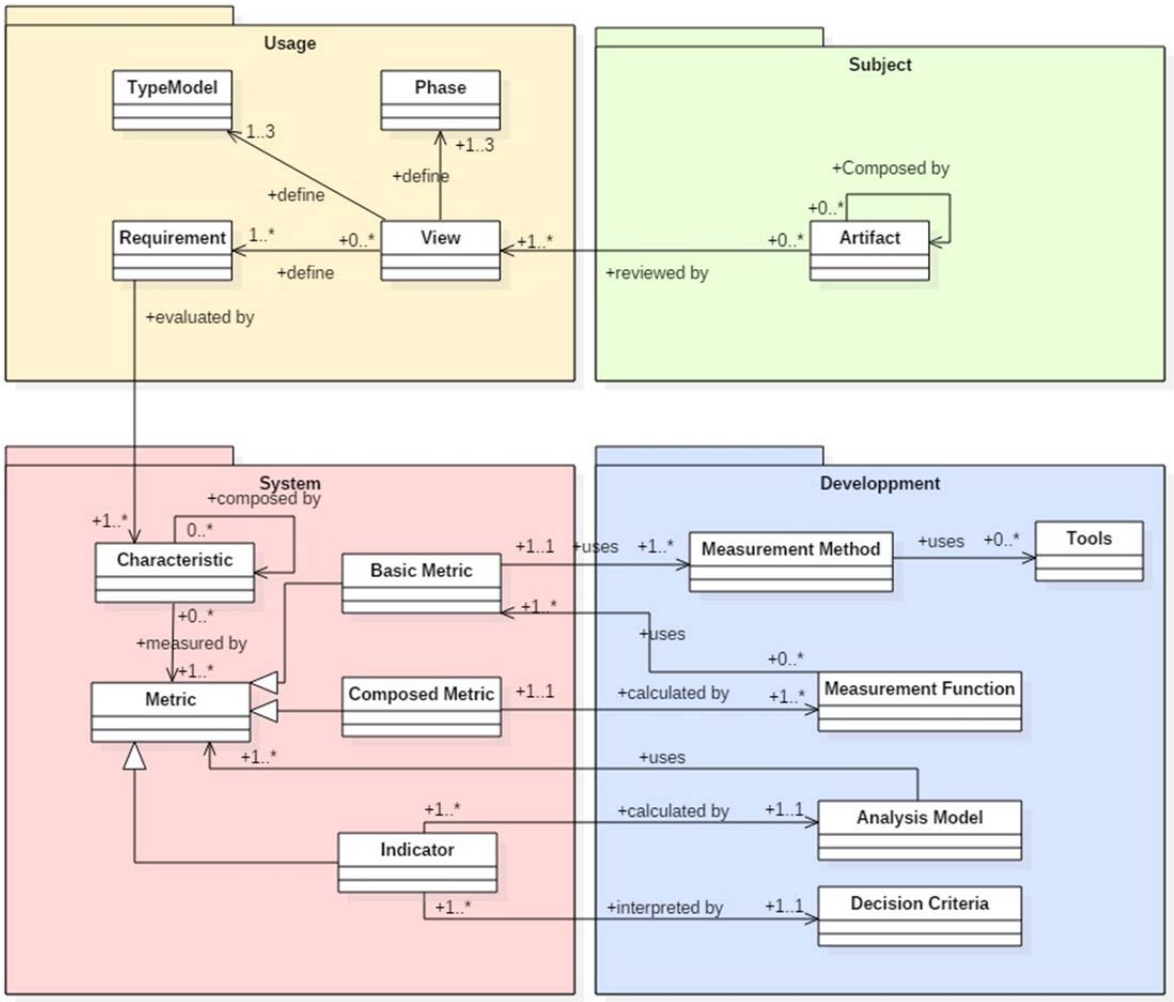

Figure 3: General Meta-model for software quality

5. CONCLUSION

Existing models of software quality are used to define a static set of characteristics and define the relationships between these different characteristics so that they should rather be selected dynamically based on stakeholder needs. The ideal solution to this problem is to have a dynamic and flexible framework, a Meta model, which will allow developers to define a quality model based on context. In this article we proposed a methodology based on the architecture model driven to build a Meta model of software quality. Using this methodology we were able to gather fundamental concepts in the field of software quality. This Meta model will overcome the gaps and limitations of existing models. It conforms to the MOF architecture and it is defined as an extension of UML Meta model. Besides, it incorporates the majority of the most important aspects of software quality in a unique framework and takes into account previous work in the field of quality software.

Our future work consists to generate an ontology of software quality. This ontology will be the basis of an integrated tool for managing software quality.

REFERENCES

[1] BOEHM, Barry W., BROWN, John R., et KASPAR, Hans. Characteristics of software quality. 1978.

[2] Dromey R. Geoff A Model for Software Product Quality [Journal]. - [s.l.] : IEEE Transactions on Software Engineering, 1995. - 2 : Vol. 21.

[3] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/INTERNATIONAL ELECTROTECHNICAL COMMISSION, *et al.* Software engineering–Product quality–Part 1: Quality model. *ISO/IEC*, 2001, vol. 9126, p. 2001.

[4] MCCALL, Jim A., RICHARDS, Paul K., et WALTERS, Gene F. *Factors in Software Quality. Volume-III. Preliminary Handbook on Software Quality for an Acquisiton Manager*. GENERAL ELECTRIC CO SUNNYVALE CA, 1977.

[5] *SHARMA*, Arun, KUMAR, Rajesh, et GROVER, P. S. Estimation of quality for software components: an empirical approach. *ACM SIGSOFT Software Engineering Notes*, 2008, vol. 33, no 6, p. 1-10.

[6] *BEHKAMAL*, Behshid, KAHANI, Mohsen, et AKBARI, Mohammad Kazem. Customizing ISO 9126 quality model for evaluation of B2B applications.*Information and software technology*, 2009, vol. 51, no 3, p. 599-609.

[7] *SRIVASTAVA*, Praveen Ranjan et KUMAR, Krishan. An approach towards software quality assessment. In : *Information Systems, Technology and Management*. Springer Berlin Heidelberg, 2009. p. 150-160.

[8] *CARVALHO*, Fernando et MEIRA, Silvio RL. Towards an Embedded Software Component Quality Verification Framework. In

: *Engineering of Complex Computer Systems, 2009 14th IEEE International Conference on*. IEEE, 2009. p. 248-257.

[9] *KUMAR*, Avadhesh, GROVER, P. S., et KUMAR, Rajesh. A quantitative evaluation of aspect-oriented software quality model (AOSQUAMO). *ACM SIGSOFT Software Engineering Notes*, 2009, vol. 34, no 5, p. 1-9.

[10] MANUEL, F. Quality Attributes for COTS Components. *I+ D Computacion, 1 (2)*, 2002, p. 128-143.

[11] GEORGIADOU, Elli. GEQUAMO—a generic, multilayered, customisable, software quality model. *Software Quality Journal*, 2003, vol. 11, no 4, p. 313-323.

[12] ORTEGA, Maryoly, PÉREZ, María, et ROJAS, Teresita. Construction of a systemic quality model for evaluating a software product. *Software Quality Journal*, 2003, vol. 11, no 3, p. 219-242.

[13] ANDREOU, Andreas S. et TZIAKOURIS, Marios. A quality framework for developing and evaluating original software components. *Information and software technology*, 2007, vol. 49, no 2, p. 122-141.

[14] GRADY, Robert B. *Practical software metrics for project management and process improvement*. Prentice-Hall, Inc., 1992.

[15] DEISSENBOECK, Florian, JUERGENS, Elmar, LOCHMANN, Klaus, *et al.*Software quality models: Purposes, usage scenarios and requirements. In :*Software Quality, 2009. WOSQ'09. ICSE Workshop on*. IEEE, 2009. p. 9-14.

[16] Adil Khammal, Youness Boukouchi, Abdelaziz Marzak, Hicham Moutachaouik, Mustapha Hain, A Reference Framework For A Classification Of Software Quality Models, IOSR Journal of Computer Engineering (IOSR-JCE), Volume 18, Issue 2, Ver. II (Mar-Apr. 2016), PP 69-76

[17] BOUKOUCHI, Youness, KHAMMAL, Adil, MARZAK, Abdalaziz, *et al*. A Meta model for Quality Software Based on the MDA Approach

[18] RUIZ, Fransisco, GENERO, Marcela, GARCÍA, Félix, *et al*. A proposal of a Software Measurement Ontology. *Department of Computer Science University of Castilla-La Mancha*, 2008.

[19] GARCÍA, Félix, RUIZ, Francisco, CALERO, Coral, *et al*. Effective use of ontologies in software measurement. *The Knowledge Engineering Review*, 2009, vol. 24, no 01, p. 23-40.

[20] *WAGNER*, Stefan, LOCHMANN, Klaus, WINTER, Sebastian, *et al*. The Quamoco quality meta-model. *Technische Universitaet München, Tech. Rep. TUMI128*, 2012.

[21] *FREY*, Alfonso Garcia, CÉRET, Eric, DUPUY-CHESSA, Sophie, *et al*.QUIMERA-Toward an Unifying Quality Meta model. In : *INFORSID 2011*. 2011.

[22] *DUBIELEWICZ*, Iwona, HNATKOWSKA, B., HUZAR, Z., *et al*. Software quality Meta model for requirement, evaluation and assessment. In : *ISIM06 Conference, vol*. 2006. p. 115-122.

[23] *MOHAGHEGHI*, Parastoo et DEHLEN, Vegard. A Meta model for specifying quality models in model-driven engineering. In : *Proc. The Nordic Workshop on Model Driven Engineering*. 2008. p. 51-65.