Extending the Conceptual Base for a Holistic Quality Evaluation Approach

Belen Rivera, Pablo Becker and Luis Olsina

GIDIS_Web, Engineering School at Universidad Nacional de La Pampa, Argentina
belenrs@yahoo.com, [beckerp, olsinal]@ing.unlpam.edu.ar

Abstract. For software organizations often performing measurement, evaluation (ME), and even change/improvement (MEC) projects, a well-established quality evaluation approach can be useful. In this direction, we have developed a *holistic quality evaluation approach* whose architecture is based on two pillars, namely: *a quality multi-view modeling framework*, and *ME/MEC integrated strategies.* In this paper, we specify the conceptual base for the former pillar. Specifically, we specify an ontology of quality views documenting its main terms, properties and relationships. Quality views are paramount for selecting evaluation strategies and strategy patterns to be assigned as resources to ME/MEC projects. Also, we show how this ontology is semantically linked with the previously built ME domain ontology.

Keywords: Ontology, Quality Views, Evaluation Strategies.

1 Introduction

For those software organizations that frequently perform quality assurance activities devoted to measurement, evaluation, and change/improvement projects, a well-founded quality evaluation approach can be useful. In this direction, we consider that counting with a *holistic quality evaluation approach* can help software organizations to reach the planning and performing of measurement, evaluation and change project goals in a systematic and disciplined way. So, clear ME/MEC project goals should be established, e.g. 'understand the usability of the XYZ mobile application'. In order to achieve this goal, a strategy with well-established activities and methods for performing ME actions should be selected. For choosing the suitable strategy from a set of strategies, the target quality view must be taken into account. A *quality view* relates accordingly an entity super-category, e.g., product, system, system in use, with a quality focus such as internal quality (IQ), external quality (EQ), and quality in use (QinU). To fulfill the project goal for the given example, the underlying quality view is the System Quality View, where System is the entity super-category to be evaluated regarding the EQ focus and the Usability characteristic.

In the last years, we have developed a *holistic quality evaluation approach* [11] whose architecture is based on two pillars, namely: (1) *a quality multi-view modeling framework*; and, (2) *ME/MEC integrated strategies.* In turn, an integrated strategy embraces the next three capabilities [2]: (i) the *ME/MEC domain conceptual base and framework*; (ii) the *process perspective specifications*; and, (iii) the *method specifications*. These three capabilities support the principle of being integrated, i.e., the same terms are consistently used in the involved activities and methods. Looking at the first capability, we have built the C-INCAMI (*Contextual-Information Need, Concept Model, Attribute, Metric and Indicator*) [12] conceptual base which explicitly and formally specifies de ME concepts, properties, relationships and constraints, in addition to their grouping into components. This domain ontology for ME was enriched with terms of the recently built process generic ontology [2]. For example, a 'measurement' -from the ME domain ontology- has the semantic of 'task' -from the process generic ontology. Likewise, the 'metric' term has the semantic of 'method'; the 'measure' has the semantic of 'outcome', and so forth. In light of having a more complete conceptual base for our *holistic quality evaluation approach*, we sought the opportunity of developing an ontology for the *quality multi-view modeling framework*, i.e., the abovementioned first pillar of our approach. Quality views are now not only formally specified in an ontology but their main terms are also linked with the C-INCAMI's non-functional requirements component.

Thus, the major contributions of this work are: (i) *Specify an ontology of quality views*; (ii) *Relate the quality view terms with the ME ontology terms*; and (iii) *Discuss its applicability* for selecting strategy patterns in ME/MEC projects.

The remainder of this paper is organized as follows. Section 2 specifies the ontology of quality views, which extends the conceptual base of our *holistic quality evaluation approach*. Section 3 stresses the practical impact of the quality multi-view framework when selecting strategy patterns for specific project goals. Section 4 describes related work and, finally, Section 5 outlines conclusions and future work.

2 Ontology of Quality Views

As commented previously, the architecture of our *holistic quality evaluation approach* is built on two pillars: *a quality multi-view modeling framework* and *ME/MEC integrated strategies*. Next, we describe the *quality multi-view modeling framework* pillar considering the proposed ontology for the domain of quality views.

The ISO 25010 standard [7] deals with quality views and quality models. It establishes 'influences' and 'depends on' relationships between quality views. However, the explicit meaning of the quality view concept is missing. Rather, it outlines quality views in the context of a system quality lifecycle model, where each view can be evaluated by means of a suitable quality model that the standard proposes. To improve this weakness, we define an ontology of quality views.

It is worthy to remark that an ontology is a way for structuring a conceptual base by specifying its terms, properties, relationships, and axioms or constraints. A well-known definition of ontology says that "*an ontology is an explicit specification of a conceptualization*" [6]. On the other hand, van Heijst *et al.* [15] distinguish different types of ontologies regarding the subject of the conceptualization, e.g., *domain ontologies*, which express conceptualizations that are intended for particular domains; and *generic ontologies*, which include concepts that are considered to be generic across many domains.

Regarding the above classification, our proposed ontology can be considered rather a domain ontology since its terms, properties and relationships are specific to the quality area. However, some terms like entity super-category can be considered generics. Fig. 1 depicts the quality views ontology using the UML class diagram [13] for representation and communication purposes. Additionally, its terms and relationships are defined in Table 1 and 2 respectively.

One core term in this ontology is *Calculable-Concept View*. This term relates the *Entity Super-Category* term with the *Calculable-Concept Focus* term. An *Entity Super-Category* is the highest abstraction level of an *Entity Category* to be characterized for measurement and evaluation purposes. On the other hand, a *Calculable-Concept Focus* is a *Calculable Concept* that represents the root of a *Calculable-Concept Model*, e.g., a quality model such as the EQ or QinU models prescribed in [7].

Fig. 1 shows that instances of *Entity Super-Category* are *Software Product*, *System*, *Process*, amongst others. On the other hand, a *Calculable-Concept Focus* for the quality domain is named *Quality Focus.* Considering other domains like the cost area, Cost Focus is other type of *Calculable-Concept Focus.* Some instances of *Quality Focus* are for example *Internal Quality*, *External Quality* and *Quality in Use*. In Table 1 we define *Internal Quality* as "*the quality focus associated to the software product entity super-category to be evaluated*", *External Quality* is defined as "*the quality focus associated to the system entity super-category to be evaluated*", and *Quality in Use* as "*the quality focus associated to the system-in-use entity super-category to be evaluated*".

The relation between an instance of a *Quality Focus* and its associated instance of an *Entity Super-Category* derives in a key concept of the ontology, viz.: *Quality View*. A *Quality View* is a *Calculable-Concept View* for quality. Instances of the *Quality View* term are *Software Product Quality View*, *System Quality View*, *System-in-Use Quality View*, *Resource Quality View* and *Process Quality View*, being all of them represented in Fig. 1. (Note that another instance is for example the *Service Quality View* which is not shown in Fig. 1).

Fig. 2 shows the *influences* and *depends on* relationships between instances of quality views which are commonly present in development, evaluation and maintenance projects. E.g., the *Resource Quality View* influences the *Process Quality View*. That is, if a development team uses a new tool or method – both considered as entities of the *Resource Entity Super-Category*- this fact impacts directly in the quality of the development process they are carrying out. In turn, the *Process Quality View* influences the *Software Product Quality View*. The *Product Quality View* influences the *System Quality*, and in turn this influences the *System-in-Use Quality View*. The *depends on* relationship has the opposite semantic. Note that more quality views than those depicted in Fig. 2 can be derived from Fig. 1. E.g., the *Process Quality View* that *influences* the *Service Quality View* could be represented. In Section 3, we discuss the utility of having well-defined quality views and relationships.

Table 1. Ontology for the domain of quality views: Term definitions.

Term	Definition
Calculable Concept (synonym: Characteristic, Dimension, Factor, Feature) (from ME ontology)	Abstract relationship between attributes of entity categories and information needs. Note 1: A *Calculable Concept*, usually called characteristic, represents a combination of measurable attributes. Therefore a characteristic can be evaluated but cannot be measured as an attribute. Note 2: A characteristic can have sub-characteristics.
Calculable-Concept Focus	Highest abstraction level of a root *calculable concept* associated to one *entity super-category* to be evaluated.
Calculable-Concept Model (from ME ontology)	The set of *calculable concepts* and the relationships between them, which provide the basis for specifying the root calculable-concept requirements and their further evaluation. Note 1: A possible instance of a *Calculable-Concept Model* is the ISO 25010 Quality-in-use Model.
Calculable-Concept View	Relationship of highest abstraction level between one *calculable-concept focus* and one *entity super-category*. Note 1: Names of *calculable-concept views* are Quality View, Cost View, among others.
Entity Category (synonym: Object Category) (from ME ontology)	Object category that is to be characterized by measuring its attributes.
Entity Super-Category	Highest abstraction level of an entity category of value to be characterized and assessed in Software Engineering organizations. Note 1: Names of entity super-categories are *Resource*, *Process*, *Software Product*, *System*, *System in use*, among others.
External Quality	It is the *quality focus* associated to the *system* entity super-category to be evaluated.
Internal Quality	It is the *quality focus* associated to the *software product* entity super-category to be evaluated.
Process	It is the *entity super-category* which embraces work definitions.
Process Quality	It is the *quality focus* associated to the *process* entity super-category to be evaluated.
Process Quality View	It is the *quality view* that relates the *process quality* focus with the *process* entity super-category.
Quality Focus	It is a *calculable-concept focus* for quality.
Quality in Use	It is the *quality focus* associated to the *system-in-use* entity super-category to be evaluated.
Quality View	It is a *calculable-concept view* for quality.
Resource	It is the *entity super-category* which embraces assets that can be assigned to processes, activities and tasks. Note 1: Examples of assets are Tool, Strategy, Software team, etc.
Resource Quality	It is the *quality focus* associated to the *resource* entity super-

	category to be evaluated.
Resource Quality View	It is the *quality view* that relates the *resource quality* focus with the *resource* entity super-category.
Software Product	It is the *entity super-category* which embraces software programs (i.e., source codes), specifications (i.e., requirements specifications, architectural specifications, data specifications, testing specifications, etc.), and other associated documentation.
Software Product Quality View	It is the *quality view* that relates the *internal quality* focus with the *software product* entity super-category.
System	It is the *entity super-category* which embraces software programs (i.e., applications) running in a computer environment, but not necessarily in the final environment of execution and usage.
System in Use	It is the *entity super-category* which embraces operative software applications used by real users in real contexts of use.
System-in-Use Quality View	It is the *quality view* that relates the *quality in use* focus with the *system-in-use* entity super-category.
System Quality View	It is the *quality view* that relates the *external quality* focus with the *system* entity super-category.

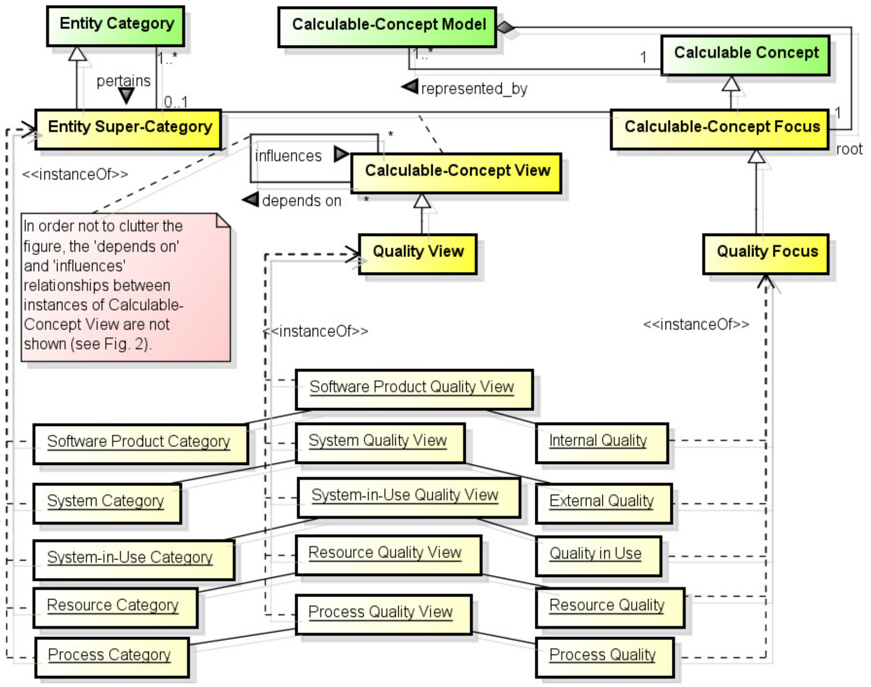

Fig. 1. Ontology for the Quality Views domain.

Table 2. Ontology for the domain of quality views: Relationship definitions.

Relationship	Definition
depends on	A *calculable-concept view* depends on other *calculable-concept view*.
influences	A *calculable-concept view* influences other *calculable-concept view*.
pertains	An *entity category* can be classified into an *entity super-category*.
represented_by	A *calculable-concept view* can be represented by one or several *calculable concept models*.

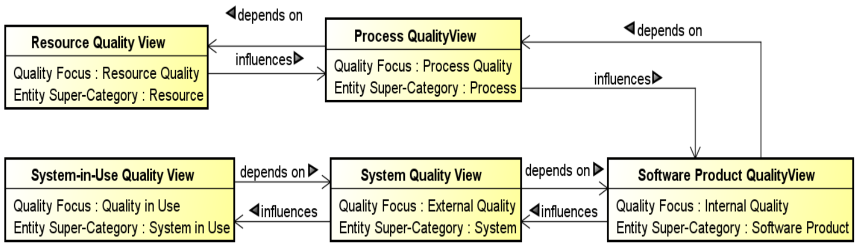
Resource Quality View
Quality Focus : Resource Quality
Entity Super-Category : Resource
Process QualityView
Quality Focus : Process Quality
Entity Super-Category : Process
System-in-Use Quality View
Quality Focus : Quality in Use
Entity Super-Category : System in Use
System Quality View
Quality Focus : External Quality
Entity Super-Category : System
Software Product QualityView
Quality Focus : Internal Quality
Entity Super-Category : Software Product
depends on
influences

Fig. 2. An instantiation of typical quality views.

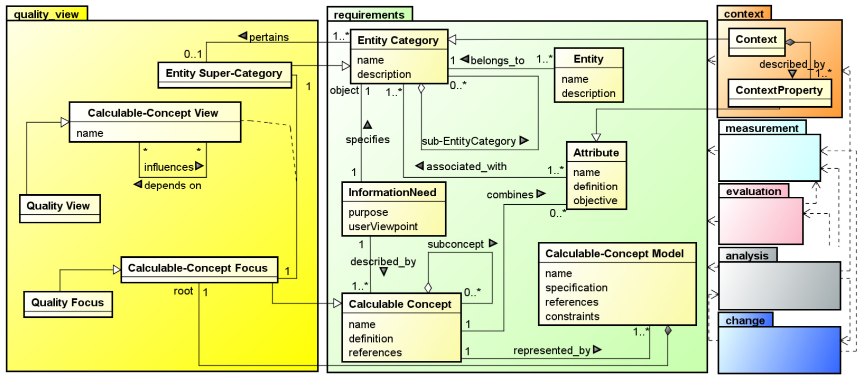

Fig. 3. The *quality_view* component which extends the C-INCAMI conceptual framework. Note that many C-INCAMI components are drawn without terms for space reasons.

The quality views ontology shares some terms with the previously developed ME ontology [12]. Particularly, an *Entity Super-Category* is an *Entity Category* –from the `requirements` component in Fig. 3-, which is defined in Table 1 as "*the object category that is to be characterized by measuring its attributes*". In turn, a *Calculable-Concept Focus* is a root *Calculable Concept* and it is *represented by* one or more *Calculable-Concept Model* –see the `requirements` component. A *Calculable-Concept Model* is defined in Table 1 as "*the set of calculable concepts and the relationships between them, which provide the basis for specifying the root calculable-concept requirements and their further evaluation*".

Ultimately, Fig. 3 shows the added `quality_view` component –which includes those yellow-colored key terms in Fig. 1- and its linking with the non-functional `requirements` component, which is one component of the C-INCAMI conceptual framework. Note also that in Fig. 1 the terms belonging to the `requirements` component are green colored as in Fig. 3.

3 Quality Views and Strategy Patterns: An Abridged Discussion

It is well-known that ontologies are widely used for different purposes [3] (e.g., natural language processing, knowledge management, information integration, semantic web processing) in different communities (e.g., knowledge engineering, web and software engineering). The previous Section has specified the ontology of quality views which is paramount for defining ME and MEC strategy patterns [14].

A strategy pattern can be seen as a general reusable solution to recurrent problems within given measurement, evaluation and change/improvement situations for specific projects' goals. So, in the following paragraphs, we analyze some strategy patterns that can be defined considering the type of ME/MEC project goal (e.g. understand, change/improve) and the type and amount of quality views that can intervene (recall Fig. 2), which can be one or more. It is worthy to remark that the quality views ontology plays a central role in defining strategy patterns. That is, without a clear specification of the terms and relationships for quality views, the ulterior specification of strategy patterns could not be done appropriately. Specifically, the quality views ontology fosters the specification and selection of appropriate strategy patterns and their instantiation regarding different ME/MEC project goals.

Usually, strategy patterns are documented by templates. In a previous work [14], we have specified a set of strategy patterns following to some extent the pattern specification template used in [5]. Our template includes the following items: (1) *name*: A descriptive and unique name, usually expressed in English; (2) *alias*: Acronym or other names for the pattern; (3) *intent:* Main objective for the pattern; (4) *motivation* (problem): Problem which solves the pattern; (5) *applicability*: Situations in which the pattern can be applied; (6) *structure* (solution): Generic structure and instantiable solution that the pattern offers; (7) *known uses*: References of real usage; (8) *scenario of use*: Concrete example and illustration for the instantiated pattern.

As above mentioned, a strategy pattern must be selected according to the type of ME/MEC project goal and the amount of involved quality views. In this sense, we distinguish at least a set of six strategy patterns. Reassuming the example commented in the Introduction, viz. 'understand the usability of the current state of the XYZ mobile application', the ME project goal has an "understand" purpose embracing the *System Quality View* (i.e., the *Entity Super-Category* is System and the *Quality Focus* is EQ, where the concrete *Entity* is the "XYZ mobile application"). Therefore, a strategy pattern that considers just one quality view for ME should be selected. In [14], this strategy pattern is the so-called *Goal-Oriented Context-Aware Measurement and Evaluation for one Quality View* (alias GOCAME_1V). Supposing by a while that the project involves also a change (MEC) goal for one quality view, it is now necessary not only to understand the current situation of the entity but also to perform changes on the entity in order to re-evaluate it and gauge the improvement gain. This strategy pattern is named *Goal-Oriented Context-Aware Measurement, Evaluation and Change for one Quality View* (alias GOCAMEC_1V). Both GOCAME_1V and GOCAMEC_1V share the same amount of involved quality views but they differ in the intended goal, i.e., while the former is intended mainly for the "understand" goal the latter is for the "improve" goal.

On the other hand, if the project involves MEC goals but for two quality views then the GOCAMEC_2V strategy pattern should be chosen. This strategy pattern addresses the fact that improving one quality view from other quality view is supported thanks to the *influences* and *depends on* relationships between quality views. As can be seen in Fig. 2, the *System Quality View influences* the *System-in-Use Quality View*, hence by evaluating and improving the *EQ Focus* of a *System* is one means for improving the *QinU Focus* of a *System in Use*. Conversely, evaluating the QinU can provide feedback to improve the EQ by exploring the *depends on* relationship. A concrete strategy derived from this pattern is the so-called SIQinU (*Strategy for Improving Quality in Use*). This strategy allows improving QinU from the EQ standpoint, as documented in the industrial case presented in [9].

The GOCAMEC_2V strategy pattern can also be instantiated for other two related quality views. For example, looking at Fig. 2 in which the resource quality (e.g., a new integrated tool) *influences* the process quality (e.g., a development process) and the process quality *depends on* the resource quality, GOCAMEC_2V should be instantiated respectively for *Resource* and *Process Quality Views*.

Furthermore, regarding the mentioned relationships between views, strategy patterns where three quality views intervene can be instantiated. For instance, we can mention the GOCAMEC_3V strategy pattern where the *Software Product*, *System* and *System-in-Use Quality Views* can be considered.

In summary, the modeling of many quality views and their relationships foster developing a family of patterns. Patterns are essentially ‘experience in a can’, to our case, ready to be opened and used by evaluators in quality assurance processes.

4 Related Work

In the literature review made about the few works that deal with the domain of quality views, we have observed there is no research defining a quality views ontology, nor an explicit glossary of terms. One of the most relevant works previously mentioned is the ISO 25010 standard [7], where different quality views and their ‘influences’ and ‘depends on’ relationships are presented informally in an annex. It illustrates that the software lifecycle processes (such as the quality requirements process, design process and testing process) influence the quality of the software product and the system; the quality of resources, such as human resources, software tools and techniques used for the process, influence the process quality, and consequently, influence the product quality; among other influences relationships between quality views. However, the explicit definition of the quality view term and the ‘influences’ and ‘depends on’ relationships are missing in its glossary. Moreover, it is not a clear association between a quality focus and an entity category nor the definitions of the different entity categories as we made in Table 1.

Other initiative related to quality views is [10] in which just the ‘influences’ relationship between EQ and QinU is determined by means of Bayesian networks, taking as reference the ISO 9126 standard [8]. However, it does not present a conceptual base in the context of a holistic quality evaluation approach as we propose.

Lastly, we can mention the 2Q2U (*Internal/External Quality, Quality in Use, Actual Usability, and User experience*) quality framework [11]. This work extends the quality models defined in [7] adding new sub-characteristics for EQ and QinU, and considers the ‘influences’ and ‘depends on’ relationships for three quality views, namely: *Software Product*, *System* and *System-in-Use Quality Views*. But an explicit ontology for the quality views domain as we propose in this paper is missing.

In summary, there are no related works for the definition and specification of an ontology of quality views. Moreover, there is no research that relates quality views' terms with non-functional requirements' terms as we documented in Section 2. However, there exists research about ontologies in software measurement, e.g., the Software Measurement Ontology (SMO) [1], in which authors relate foundational ontologies with domain ontologies. This clear separation of concern between generic and domain ontologies will be dealt in a future for our ontology of quality views.

Finally, having well-defined quality views and their relationships provides the basis for a more robust selection of strategy patterns for ME/MEC project goals (as commented in Section 3) and also contributes to enhance our quality evaluation approach.

5 Conclusions and Future Work

As commented in the Introduction Section, the architecture of our *holistic quality evaluation approach* is built on two columns: (1) *a quality multi-view modeling framework,* and (2) *ME/MEC integrated strategies.* One discussed contribution in this work is the specification of the ontology of quality views, aimed at adding robustness to our approach. To build this ontology we have reviewed the related literature to the quality views domain. Specifically, we have observed that there is no such an ontology, taxonomy or glossary for this domain.

Note that in this paper, we have addressed the ontology representation and a possible instantiation of it rather than the ontology construction process itself. Nevertheless, the stages proposed in the METHONTOLOGY [4] approach were followed such as specification, conceptualization, formalization and integration. The integration stage was done by relating the quality views ontology with the previous C-INCAMI's ME ontology. This fulfills the second contribution stated in the Introduction Section. As a consequence, the former conceptual base for the *holistic quality evaluation approach* was enhanced.

Regarding the third stated contribution, we have analyzed in Section 3 the importance of having well-defined quality views and their relationships with the aim of defining and selecting strategy patterns for different ME/MEC project goals.

As future work, we envision the development of a strategy pattern recommender system as a practical use of the quality views ontology in the context of the holistic quality evaluation approach. This system can be useful when an organization establishes a ME/MEC project goal. So, taking into account the type of project goal and the amount of involved quality views, the strategy pattern recommender system will suggest the suitable strategy pattern that fits that goal.

References

1. Barcellos M.P., Falbo R. A., Dalmoro R.: A Well-Founded Software Measurement Ontology. In 6th International Conference on Formal Ontology in Information Systems (FOIS), pp. 213-226, (2010)
2. Becker P., Papa F., Olsina L.: Process Ontology Specification for Enhancing the Process Compliance of a Measurement and Evaluation Strategy. In CLEI Electronic Journal 18(1), pp. 1-26. ISSN 0717-5000, (2015)
3. Corcho O., Fernández-López M., Gómez-Pérez A.: Methodologies, tools and languages for building ontologies. Where is their meeting point? Data & Knowledge Engineering 46(1), pp. 41–64, (2003)
4. Fernández-López M., Gómez-Pérez A., Juristo N.: METHONTOLOGY: From Ontological Art Towards Ontological Engineering. Spring Symposium on Ontological Engineering of AAAI, pp. 33–40, Stanford University, California, (1997)
5. Gamma E., Helm R., Johnson R., Vlissides J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addisson-Wesley, ISBN 0-201-63361-2, (1995)
6. Gruber T.R.: A Translation Approach to Portable Ontologies. Knowledge Acquisition, 5(2), pp. 199–220, (1993)
7. ISO/IEC 25010: Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, (2011)
8. ISO/IEC 9126-1: Software Engineering Product Quality - Part 1: Quality Model (2001)
9. Lew P., Olsina L., Becker P., Zhang, L.: An Integrated Strategy to Systematically Understand and Manage Quality in Use for Web Applications. Requirements Engineering Journal, Springer London, 17(4), pp. 299-330, (2012)
10. Moraga M.A, Bertoa M.F., Morcillo M.C., Calero C., Vallecillo A.: Evaluating Quality-in-Use Using Bayesian Networks. In QAOOSE 2008, Paphos, Cyprus, pp 1-10, (2008)
11. Olsina L., Lew P., Dieser A., Rivera M.B.: Updating Quality Models for Evaluating New Generation Web Applications. In Journal of Web Engineering, Special issue: Quality in new generation Web applications, Abrahão S., Cachero C., Cappiello C., Matera M. (Eds.), Rinton Press, USA, 11(3), pp. 209-246, (2012)
12. Olsina L., Papa F., Molina H.: How to Measure and Evaluate Web Applications in a Consistent Way. HCIS Springer book *Web Engineering: Modeling and Implementing Web Applications*; Rossi G., Pastor O., Schwabe D., and Olsina L. (Eds.), pp. 385-420, (2008)
13. OMG-UML. Unified Modeling Language Specification, Version 2.0. (2005)
14. Rivera M.B., Becker P., Olsina L.: Strategy Patterns for Measurement, Evaluation And Improvement Projects (In Spanish). XVIII Iberoamerican Conference in Software Engineering (CIbSE'15), Lima, Perú, pp. 166-180, ISBN: 978-9972-825-80-4, (2015)
15. van Heijst G., Schreiber A.T., Wielinga B.J.: Using Explicit Ontologies in KBS Development. International Journal of Human-Computer Studies, 46, pp.183-292, Academic Press, Inc. Duluth, MN,USA, (1997)