Journal of Web Engineering, Vol. 15, No.5&6 (2016) 433-464

QUALITY VIEWS AND STRATEGY PATTERNS FOR EVALUATING AND IMPROVING QUALITY: USABILITY AND USER EXPERIENCE CASE STUDIES

BELEN RIVERA, PABLO BECKER, LUIS OLSINA
GIDIS_Web, School of Engineering, National University of La Pampa
General Pico, La Pampa 6360, Argentina
riveramb@ing.unlpam.edu.ar, *beckerp@ing.unlpam.edu.ar*, *olsinal@ing.unlpam.edu.ar*

Received July 13, 2015
Revised February 23, 2016

Nowadays, software and web organizations are immersed in very competitive markets. This situation challenges organizations for paying special attention to the quality of applications and services offered to consumers. For those that frequently carry out quality assurance activities devoted to measurement, evaluation (ME) and change/improvement (MEC) projects, well-founded quality evaluation and improvement approaches can be a key competitive issue. In this direction, we have developed an integrated quality approach whose architecture is based on quality views, and ME/MEC strategies. In order to bolster the former aspect, we specify, in this work, an ontology for quality views. Quality views and their 'influences' and 'depends on' relationships between them, are paramount for defining and selecting evaluation and improvement strategy patterns and ultimately specific strategies to be used in ME/MEC projects. A strategy pattern is a reusable solution to recurrent problems in MEC projects. For a project goal, the selected strategy pattern allows one to instantiate a specific strategy, which embraces a set of tailored activities and methods for measurement, evaluation, analysis and change for improvement. Also we discuss a set of strategy patterns and document two patterns, which were used in two case studies for understanding and improving Usability and User Experience.

Key words: Quality Views, Strategy Patterns, Evaluation, Improvement, Usability.

Communicated by: J-G Houben & J. Vanderdonck

1 Introduction

Nowadays, software and web organizations are immersed in very competitive markets. For developing, maintaining and managing successful software and web applications, quality and economic issues should be considered, among other important competitive factors. For leveraging these factors in a systematic and disciplined way, organizations should perform quality assurance activities and adopt well-established quality evaluation and improvement approaches for fulfilling measurement, evaluation (ME) and change (MEC) project goals. For instance, a quality evaluation approach should clearly establish *ME/MEC strategies* which are able to specify quality characteristics and attributes regarding one or more *quality views*, and ultimately use metrics and indicators and their

values for analysis, recommendations and improvement activities, in a trustworthy manner [8, 23].

As highlighted above, quality views is a key concept of an evaluation approach. Although quality views have been characterized since early 80's, very often they have been weakly specified. For example, in 1984, Garvin [16] has described quality from five different views: 1) *Transcendental view*: Quality, as synonymous with "innate excellence", is something we can recognize but not define; 2) *User view*: This is a personal, subjective view of quality, which lies in the eyes of the beholders; i.e., quality is fitness for purpose; 3) *Product view*: Quality is tied to inherent product characteristics and attributes; 4) *Manufacturing view*: Quality is conformance to specifications and requirements; and 5) *Value-based view*: Quality depends on the amount the customer is willing to pay for it. Garvin points out that there is a "*need to actively shift one's approach to quality as products move from design to market. The characteristics that connote quality must first be identified through market research (a user-based approach to quality); these characteristics must then be translated into identifiable product attributes (a product-based approach to quality); and the manufacturing process must then be organized to ensure that products are made precisely to these specifications (a manufacturing-based approach to quality). A process that ignores any one of these steps will not result in a quality product. All three views are necessary and must be consciously cultivated*".

More recently, ISO standards [21, 22] and other authors such as [5, 30] consider also some quality views and the influence relationship between them. For example, Baskerville *et al.* [5] state "*the product-based quality view says that because quality is linking to measurable product features, product quality is conformance to stated product requirements* [...]. *A user-based view, in contrast, says that quality is about fulfilling used expectations, or 'fitness for purpose'*.". In 2001, ISO 9126-1 [22] made explicit the 'influences' and 'depends on' relationships between views by indicating that "*Process quality* [...] *contributes to improving product quality, and product quality contributes to improving quality in use. Therefore, assessing and improving a process is a means to improve product quality, and evaluating and improving product quality is one means of improving quality in use. Similarly, evaluating quality in use can provide feedback to improve a product, and evaluating a product can provide feedback to improve a process.*". Also, it states that the assessment of these views can be achieved by using process measures in addition to internal, external and quality in use measures respectively.

In 2011, the ISO 25010 standard (which superseded to [22]), adds the resource view, maintaining the hypothesis of 'influences' and 'depends on' (or 'is determined by') relationships. It states 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. Software product quality, as well as the quality of other components of a system, influences the quality of the system. The system quality has various influences (effects) depending on the contexts of use. The context of use can be defined by a set of a user, a task, and the environment.*". Figure 1 illustrates the target entities for quality and their relationships as per ISO.

Also it is important to remark that the CMMI [9] *de facto* standard is supported by the evidence that assessing and improving the process (i.e., by fulfilling accordingly the required specific and generic goals of process areas) is a means to improve product and service quality.

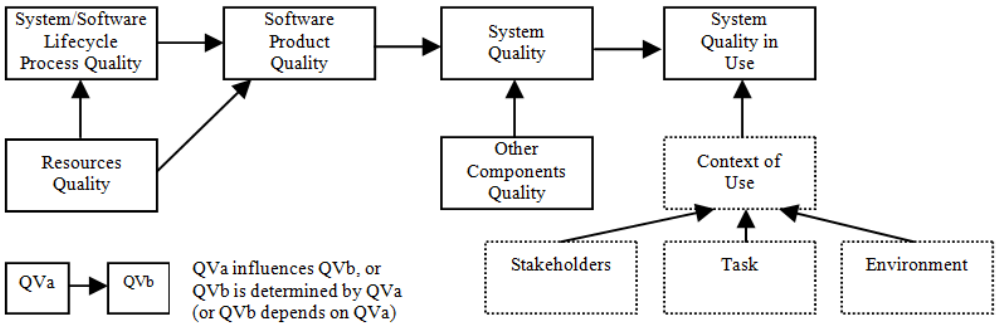

Figure 1: Target entities and their relationships for evaluating quality (adapted from [21]. Note that QV means Quality View)

Recently, we have developed an evaluation approach that includes quality views as well. Furthermore, our approach considers also ME/MEC strategies for achieving project goals. Let's introduce the approach with the following example. The ME project goal is 'Understand the Facebook mobileapp usability weaknesses'. For achieving this goal, one specific ME strategy with well-established activities and methods should be selected.

Strategy is a frequently used and broad term, so for our purposes, we have defined it as: "*principles, patterns, and particular domain concepts and framework that may be specified by a set of concrete processes, in addition to a set of appropriate methods and tools as core resources for helping to achieve a project goal*" [6]. In a broad sense, for choosing the suitable strategy from a set of ME/MEC strategies, one or more *quality views* must be taken into account. A quality view relates an *entity super-category* such as product, system, system in use, accordingly with a *quality focus* such as internal quality, external quality, and quality in use (QinU).

In the goal statement of the above example, the underlying quality view is the System Quality View, where System is the entity super-category to be evaluated regarding the External Quality focus and the Usability characteristic. Note that Facebook mobile app is the concrete entity for the System entity super-category. In turn, Usability is represented in an external quality model, which may combine sub-characteristics (e.g., Operability) and attributes (e.g., Stability of main controls). The ME strategy should allow selecting metrics for quantifying attributes and indicators for interpreting these quality requirements, with the aim to analyze, recommend and propose change actions on the basis of the yielded outcomes (i.e., measures and indicators values).

In summary, we have developed a *holistic quality evaluation and improvement approach* [34] 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 three capabilities: (2.i) the *ME/MEC domain conceptual base and framework*; (2.ii) the *process perspective specifications*; and, (2.iii) the *method specifications*. These three capabilities support the principle of being integrated [37] since for instance the same terms are consistently used for activities and methods that the strategy prescribes.

Looking at the (2.i) capability of a strategy, we have built a conceptual framework so-called C-INCAMI (*Contextual-Information Need, Concept model, Attribute, Metric and Indicator*) [35], which

explicitly and formally specifies the ME concepts, properties, relationships and constraints, in addition to their grouping into components. This domain ontology for ME was also enriched with terms of a process generic ontology [6]. 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, and considering the characterization made on quality views in the above-mentioned research, we sought the opportunity of developing an ontology for quality views for the (1) pillar of our evaluation approach. Despite ISO 25010 deals with quality views, an explicit definition of the quality view term and other related terms are missing in this standard. Also, other related work mix up the quality view concept with the quality focus concept, as we will discuss later on. Consequently, developing an ontology of quality views can be helpful to provide an explicit semantic for this domain, which can benefit the semantic processability of goal statements for selecting the suitable strategy for a specific project. These are the major requirements for building such an ontology.

Furthermore, quality views and their 'influences' and 'depends on' relationships are paramount for defining ME and MEC strategies. Some known ME/MEC strategies for software are: *Goal Question Metric* (GQM) [3], *Continuous Quality Assessment Methodology* (CQA-Meth) [38], *Practical Software Measurement* (PSM) [28], and *Quality Improvement Paradigm* (QIP) [4], amongst others. However, most of these strategies have not well specified some of the three capabilities, i.e., the ME domain conceptual base and framework (2.i), the process perspective specifications (2.ii), or the method specifications (2.iii). Nor are these capabilities often considered simultaneously, in an integrated way. Moreover, in the quoted ME/MEC strategies quality views and their relationships have often been neglected.

In the last decade, we have earned experience in developing a couple of specific ME/MEC strategies. We have developed the GOCAME (*Goal-Oriented Context-Aware Measurement and Evaluation*) and the SIQinU (*Strategy for Improving Quality in Use*) strategies, which were applied in several specific evaluation and improvement projects [26, 34-37]. For these ME/MEC projects, one or two quality views were considered. Also, both strategies have the three above-mentioned capabilities, which are supported in an integrated way.

Lately, we have envisioned the idea of packaging the earned experience into *strategy patterns*. It is recognized that patterns have had and continue to have a significant impact in software and web engineering [13, 15]. In a nutshell, the pattern's main aim is to provide a general and reusable solution to a recurrent problem. We have observed that strategy patterns can be applied to recurrent ME or MEC problems of any project. As a result, we specify a set of strategy patterns that offers flexible and tailorable solutions for evaluating and improving the quality focuses for different entities in ME/MEC projects, considering one or more quality views. To the best of our knowledge, this specific contribution fills a gap in the current literature.

Therefore, the major contributions documented in this paper are: (i) Specify an ontology of quality views; (ii) Analyze strategy patterns for different quality views and project goals; and (iii) Specify two concrete strategy patterns and perform their instantiation. We illustrate one strategy pattern for improving one quality view using the Facebook mobileapp case study. For the other pattern, we

exemplify fragments of the JIRA[a] webapp case study, for improving the User eXperience (UX) considering two quality views and the 'influences' and 'depends on' relationships.

Following this introduction, Section 2 describes related work addressing research that deals with quality views and strategy patterns. Section 3 specifies the ontology of quality views, in the context of our evaluation and improvement approach. Also, quality views and the external quality and QinU focuses are illustrated considering the Usability and UX characteristics. Section 4 documents thoroughly two ME/MEC strategy patterns, and stresses the practical impact of the quality multi-view framework when defining and selecting strategy patterns for specific project goals. Section 5 discusses other strategy patterns and their usefulness. Finally, Section 6 draws our main conclusions and outlines future work.

2 Related Work

In this work, we present a *holistic quality evaluation and improvement approach* whose architecture is based on two pillars, namely: (1) *a quality multi-view modeling framework*; and, (2) *ME/MEC integrated strategies*. First, regarding the state-of-the-art literature we analyze the research work related to ontologies of quality views. Second, we review those works that deal with ME/MEC strategies and ultimately with strategy patterns. Also, we discuss if the existing research about a holistic quality evaluation approach takes into account these two concerns (pillars) in an intertwined and integrated manner.

Regarding the first pillar, there exists literature as quoted in the Introduction Section that deals with quality views. But as far as we know there is no research defining and specifying an ontology of quality views, nor an explicit glossary of terms. In addition to the seminal Garvin work [16], one of the most relevant documents previously cited is the ISO 25010 standard, in which different quality views and their 'influences' and 'depends on' relationships are represented informally in its Annex C. (Note that these cause-effect relationships between views have been to a some extent empirically observed [9, 14, 26, 29, 30], though more empirical research in this area is still needed). However, in ISO [21] the explicit meaning of the quality view concept is missing. Moreover, there is no clear association between a quality focus and an entity category, nor explicit definitions of the different entity categories as we do in Table 1. Rather, it outlines views in the context of a system quality lifecycle model (Figure 1), where some views can be evaluated by means of the quality model that the standard proposes.

Another initiative related to quality views is analyzed in [29] in which just the 'influences' relationship between external quality and QinU characteristics is determined by means of Bayesians networks, taking as reference the ISO 9126-1 [22] standard. However, it does not discuss a holistic evaluation approach that links quality views with ME/MEC strategies, as we propose. Finally, in [34] the 2Q2U (*internal/external Quality, Quality in use, actual Usability, and User experience*) quality framework is proposed. This framework extends the quality models defined in [21] adding new sub-characteristics for external quality 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 the explicit definition of quality views terms as we propose in this paper is missing. Also, the 2Q2U quality models were instantiated using an integrated strategy called SIQinU [26]. This strategy allows improving QinU incrementally, from changes made on the system. SIQinU is an

[a] www.atlassian.com/software/jira/

instance of one of the strategy patterns that we document in Section 4.2.

Regarding the second concern, i.e., ME and MEC integrated strategies, there exists a couple of related works ([2, 4, 28, 38], etc.). Specifically, [2] presents GQM$^+$Strategies, which is built on top of the so-called GQM strategy [3]. Both strategies include the principle of the three integrated capabilities [37]. But none consider the quality views' concepts and the 'influences' and 'depends on' relationships, nor the ME/MEC strategy pattern idea. In [27], measurement patterns are defined to establish objectives, sub-objectives and metrics for an organizational goal starting from the GQM approach. The intention of these patterns is to give reusable solutions to similar problems found in the creation of measurement programs. Additionally, authors state that the idea of measurement patterns was taken from [15], but the specification of the illustrated patterns follows no recommended style such as name, intention, problem, solution/structure, known uses, etc.

Many researches that deal with patterns are very often intended for early stages of development and change, focusing for instance on usability patterns and user interface designs, or architectural designs. But, they are seldom intended for evaluation and improvement stages in which quality views and MEC strategy patterns should be used appropriately. For example, authors in [13, 14] define a framework that expresses relationships between Software Architecture and Usability. The proposal consists of an integrated set of design solutions that have been identified in various practical cases in industry. But in our opinion, a clear separation of concerns among quality views, quality models, ME/MEC integrated strategies and strategy patterns is missing.

In summary, there is no related work for the definition and specification of an ontology of quality views. Additionally, there is no research that links the quality views' terms with non-functional requirements' terms as we will document in Section 3.1.1. Our approach ties together quality views (entity categories and quality focuses) and their relationships, in addition to customizable strategies for measurement, evaluation, analysis and improvement, which can be packaged into strategy patterns. Strategy patterns are aimed at easing the strategy instantiation for common and recurrent ME/MEC projects' goals.

3 Foundations for the Holistic Quality Evaluation and Improvement Approach

As introduced above, the architecture of our evaluation approach is built on two pillars. Sub-section 3.1, discusses the first pillar, that is, the *quality multi-view modeling framework*, which specifies the proposed ontology of quality views and the grouping of its concepts into the `quality_view` component. This ontology allows specifying for instance Resource, Product, System, and System-in-Use Quality Views, which are paramount for defining strategy patterns. Sub-section 3.2, analyzes what is an integrated strategy for the purpose of evaluation and improvement.

3.1 Quality Multi-View Modeling Framework

A ME/MEC project can involve one or more entity super-categories such as Software Product, System, System in use. Each entity super-category is evaluated considering its corresponding quality focus such as internal quality, external quality, and QinU. The relationship between an entity super-category and its quality focus is called Quality View. For the Facebook example, the System entity super-category and the External Quality focus conform the System Quality View. Also for each quality view an

appropriate quality model must be instantiated, as part of the definition and evaluation of non-functional requirements for a ME project. A quality model has a quality focus, which is the root characteristic (External Quality) in addition to characteristics and sub-characteristics (Usability and Operability) to be evaluated which combine measurable attributes (Stability of controls). So the *quality multi-view modeling framework* embraces concepts such as quality view, quality model, relationships between quality views, among other issues.

Next, sub-section 3.1.1 shows the ontology of quality views and the linking of the `quality_view` component with the previously developed C-INCAMI conceptual framework [6, 35]. Then, sub-section 3.1.2 discusses, for the sake of exemplification, how Usability and UX characteristics can be related with quality views.

3.1.1 Ontology of Quality Views

The cited ISO 25010 standard deals with quality models and to a lesser extent with quality views. It establishes 'influences' and 'depends on' relationships between quality views, but, as commented in Section 2 the explicit meaning of the quality view and quality focus terms and other related terms, as well as the linking with non-functional requirement terms such as Information Need are missing. In order to improve these weaknesses and to fulfill the requirements indicated in the Introduction Section, we have defined an ontology of quality views.

An ontology is a way of 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*" [17]. On the other hand, van Heijst *et al.* [39] distinguish different types of ontologies regarding the subject of the conceptualization such as domain and generic ontologies. Regarding this classification, the quality views 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 generic [18].

Figure 2 depicts the quality views ontology using the UML class diagram [32] for representation and communication purposes. Additionally, its terms and relationships are defined in Table 1 and Table 2 respectively. For the construction of the ontology, we have followed the stages proposed in the METHONTOLOGY [12] approach. Nevertheless, it is not the aim of this paper addressing the ontology construction process itself. Instead, we present the ontology representation and a possible instantiation of it.

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. In turn, a Calculable-Concept Focus is a *Calculable Concept* that represents the *root* of a *Calculable-Concept Model*.

Figure 2 shows that instances of Entity Super-Category are *Software Product*, *System*, and *Process*, amongst others. On the other hand, a Calculable-Concept Focus can be for example a *Quality Focus* or a *Cost Focus*. Note that Cost Focus and *Cost View* are not directly related with the quality domain, so they are gray-colored terms in Figure 2 and are not defined in Table 1. Some instances of Quality Focus are for example *Internal Quality*, *External Quality* and *Quality in Use*.

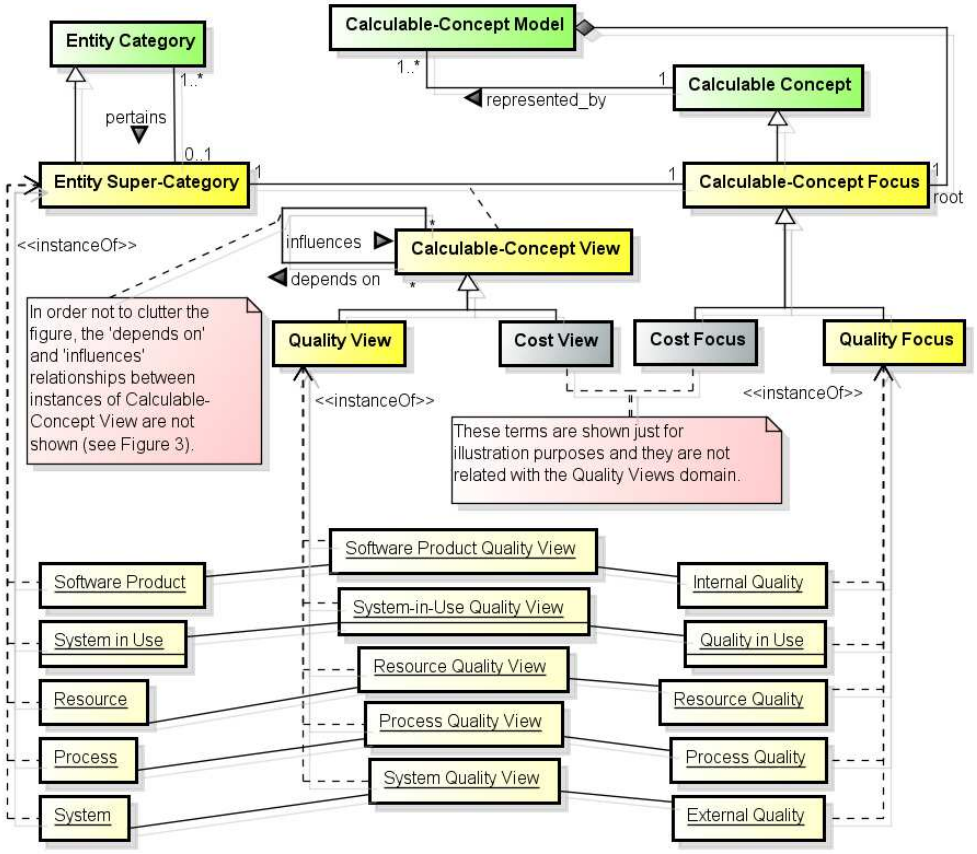

Figure 2: Terms and some instances for the ontology of Quality Views.

Table 1: Ontology of quality views: Term definitions.

Term	Definition
Calculable Concept (synonym: Characteristic, Dimension, Factor, Feature) (from ME ontology)	A characteristic that represents a combination of measurable attributes. Note 1: A *calculable concept* can be evaluated but cannot be measured as an attribute -at least in a non very trivial way such as "good" or "bad". Note 2: A characteristic can have sub-characteristics.
Calculable-Concept Focus	It is a *calculable concept* which represents the root of a *calculable-concept model*. Note 1: A *calculable-concept focus* is 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 non-functional 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	Abstract relationship 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	Object category that is to be characterized by measuring its attributes.

(synonym: Object Category) (from ME ontology)	
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 (e.g., source codes), specifications (e.g., 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 running in a computer environment (e.g., applications), 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.

Table 2: Ontology of quality views: Relationship definitions.

Relationship	**Definition**
dependsOn	A *calculable-concept view* depends on other c*alculable-concept view*.
describes	A *ME information need* describes a *calculable-concept focus.*
influences	A *calculable-concept view* influences other *calculable-concept view.*
isRepresentedBy	A *calculable-concept focus* can be represented by one or several *calculable-concept models.*
pertains	An *entity categ*ory can be classified into an *entity super-category*.

The relationship between an Entity Super-Category and its associated Quality Focus is the *Quality View* key concept. A Quality View is a Calculable-Concept View for quality. Instances of the Quality View term are *Resource Quality View*, *Process Quality View*, *Software Product Quality View*, *System Quality View*, and *System-in-Use Quality View* terms, as shown in Figure 2. It is worth mentioning that in the figure not all instances of quality views are shown (e.g., the Service Quality View).

Compared with the Garvin characterization, we can observe some matching between views. For example, our System-in-Use Quality View corresponds totally to the Garvin's User View. Note that Garvin analyzes and exemplifies the manufacturing and product views for industries other than software, so we cannot establish a straightforward mapping. However, our System Quality View matches totally with the ISO 25010's System Quality and to some extent to the Garvin's Product View. Also, the Manufacturing View has some correspondence with our Software Product Quality View (software is a kind of product), likewise to the ISO's Software Product Quality (see Figure 1). Garvin's Value-based View is not present in ISO. Nevertheless, we could model it considering the Quality and Cost Views depicted in Figure 2.

Figure 3, which mimics to some extent Figure 1, shows the *influences* and *depends on* relationships between instances of quality views which are commonly present in development, evaluation and maintenance projects. Thus, the Resource Quality View influences the Process Quality View. For example, 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. Likewise, the Process Quality View influences the Software Product Quality View. The Product Quality View influences the System Quality, and this in turn influences the System-in-Use Quality View. Conversely, the depends on relationship has the opposite semantic.

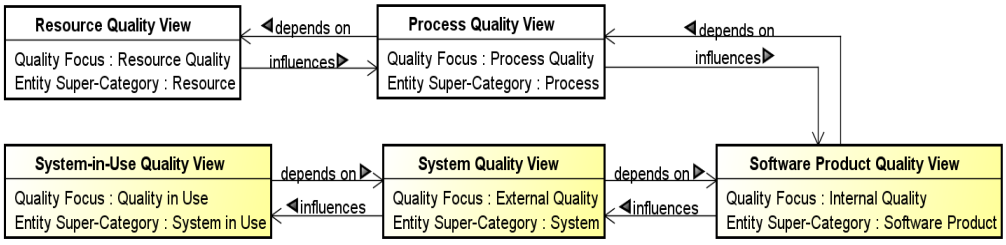

Figure 3: An instantiation of typical quality views in software development projects.

Also Figure 4 shows the corresponding *dependent* and *independent view* roles for the influences and depends on relationships. For example, in Figure 3, the Resource Quality View is the *independent view* while the Process Quality View is the *dependent view*. Note also that these relationships are transitive; e.g., the Resource Quality View influences the Software Product Quality View, as represented by ISO (see Figure 1).

On the other hand, it is important to address that the quality views ontology shares some terms with the ME ontology, which was presented in [35]. For example, an *Entity Super-Category* is an *Entity Category*, which is a term from the `non-functional requirements` component. As a result, in Figure 4, the analyzed quality view terms are grouped into the `quality_view` component which are linked with the former C-INCAMI `non-functional requirements` component.

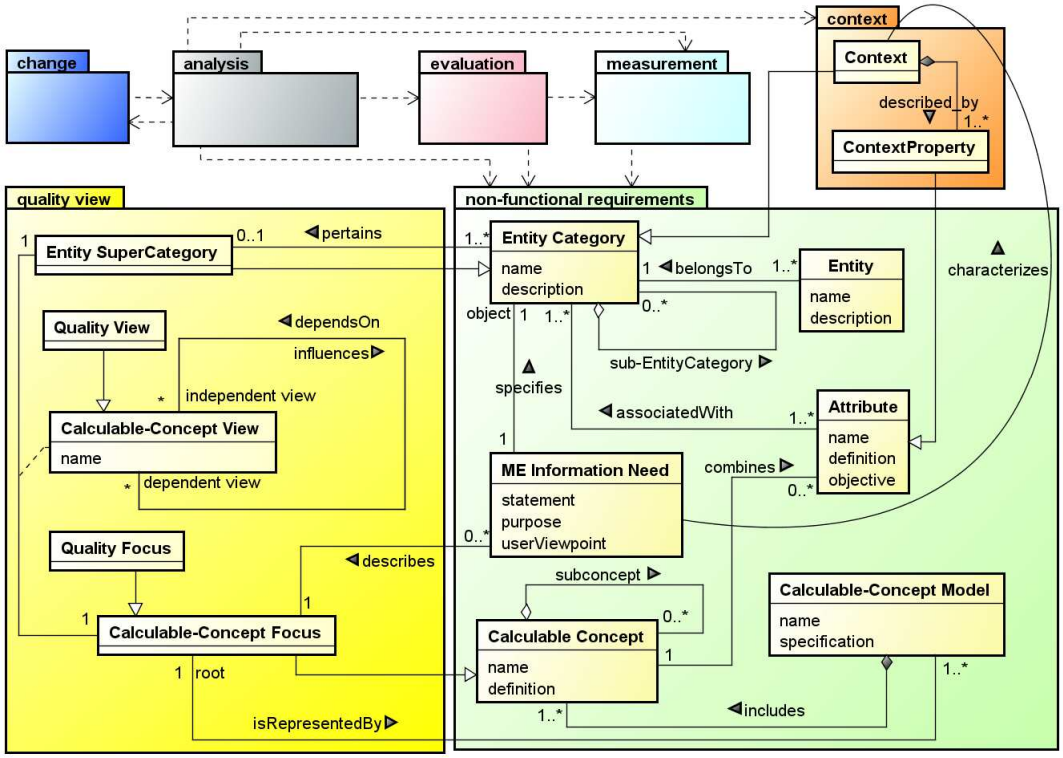

Figure 4: The quality_view component which extends the C-INCAMI conceptual framework. (Note that many C-INCAMI components are drawn without terms for better visualization. In Figure 9, the measurement and evaluation components are expanded).

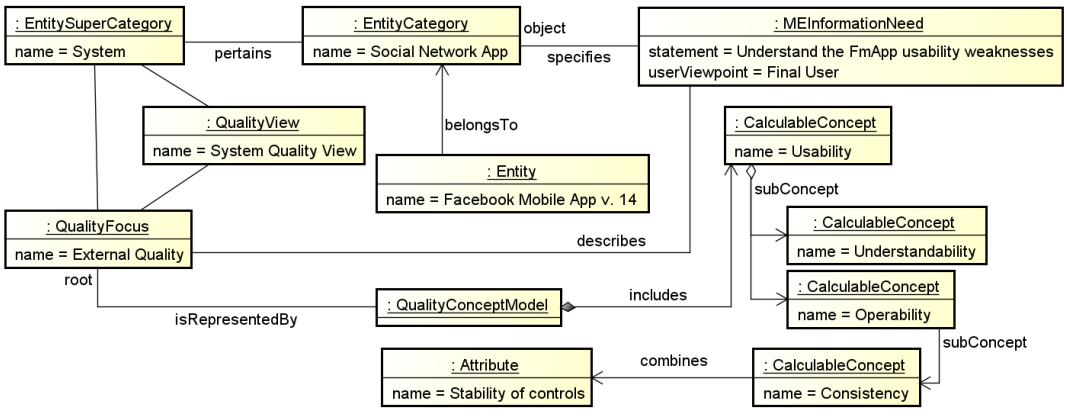

Figure 5: Instantiation of terms for the *quality_view* and *non-functional requirements* components. (Note: FmApp means Facebook mobile app).

Figure 5 shows the instantiation of terms for the quality_view and non-functional requirements components, following the example of the Introduction Section. In this case, the ME project goal is operationalized in a *ME Information Need* whose statement is 'Understand the Facebook mobileapp usability weaknesses'.

3.1.2 Relating Usability and UX Characteristics with Quality Views

Usability and UX have been widely discussed in many publications such as [7, 13, 19, 20, 24, 31], just to mention a few. When evaluating Usability and UX for mobile and web entities, regarding the project goal suitable quality views should be considered [20, 33]. For this aim, potential quality views are those yellow-colored in Figure 3, namely: Software Product, System and System-in-Use Quality Views. For each quality focus of a quality view, a quality model should be selected. A quality model specifies non-functional requirements in the form of characteristics, sub-characteristics and attributes.

A question that a reader might ask himself is: to which quality focus can Usability and UX characteristics be related? In short, looking at the quoted usability literature, Usability can be related to the Internal Quality and External Quality focuses, while UX to the Quality in Use focus.

In Figures 6 and 7, the Usability, Actual Usability (Usability in use, as synonym), and UX characteristics are defined. These characteristics are included in the 2Q2U v2.0 quality models [33, 34], which are basically an extension of the ISO 25010 models [21].

Taking into account the Usability definition in Figure 6, we observe that considers Software Product and System entity super-categories, so Usability and its sub-characteristics (i.e. Understandability, Learnability, Operability, User error protection, UI aesthetics and Accessibility) are related to the Internal Quality and External Quality focuses respectively.

External Quality (root of the *External Quality Model*)

- ***Usability***: Degree to which the product or system has attributes that enable it to be understood, learned, operated, error protected, attractive and accessible to the user, when used under specified conditions.
  - ***Understandability*** (synonym *Appropriateness Recognizability*): Degree to which users can recognize whether a product or system is appropriate for their needs. Note: Same ISO 25010 definition.
  - ***Learnability***: Degree to which the product or system enables users to learn its app.
  - ***Operability***: Degree to which a product or system has attributes that make it easy to operate and control. Note: Same ISO 25010 definition
  - ***User Error Protection***: Degree to which a product or system protects and prevents users against making errors and provides support to error tolerance.
  - ***UI Aesthetics*** (synonym *Attractiveness*): Degree to which the UI enables pleasing and satisfying interaction for the user. Note: Same ISO 25010 definition.
  - ***Accessibility***: Degree to which a product or system can be used by people with the widest range of characteristics and capabilities for an intended user goal.

Figure 6: Definitions of sub-characteristics related to the Usability characteristic for the External Quality focus.

Quality in Use (root of the *Quality in Use Model*)

- ***Actual User Experience***: Degree to which a system in use enable specified users to meet their needs to achieve specific goals with satisfaction, actual usability, and freedom from risk in specified contexts of use
- ***Actual Usability*** (synonym *Usability in use*): Degree to which specified users can achieve specified goals with effectiveness, efficiency, learnability in use, and without communicability breakdowns in specified contexts of use
  - ***Effectiveness***: Degree to which specified users can achieve specified goals with accuracy and completeness in specified contexts of use
  - ***Efficiency*** (**in use**): Degree to which specified users expend appropriate amounts of resources in relation to the effectiveness achieved in specified contexts of use
  - ***Learnability*** (**in use**): Degree to which specified users can learn efficiently and effectively while achieving specified goals in specified contexts of use
- ***Satisfaction***: Degree to which user needs are satisfied when a system is used by specified users in a specified context of use

Figure 7: Definitions of some characteristics and sub-characteristics for the QinU focus.

On the other side, regarding the Actual UX definition in Figure 7, we observe that considers the System-in-Use entity super-category, so UX and its sub-characteristics (i.e. Satisfaction, and Actual Usability, etc.) are related to the Quality in Use focus.

As a consequence, Usability and UX are linked mainly to the three yellow-colored quality views represented in Figure 3. That is, Usability is a characteristic related to both Software Product Quality View and System Quality View. Instead, UX is a characteristic related only to the System-in-Use Quality View. This clear separation of concerns between Usability concepts and quality views as well as their 'influences' and 'depends on' relationships foster a more robust evaluation and improvement approach, as we discuss later on.

Figure 8 illustrates two target entity super-categories (System and System in Use) and their corresponding quality focuses (External Quality and QinU) with some quality characteristics. This rough schema is derived from some components of Figure 4. For the System entity super-category (System box in Figure 8) other sub-entity categories are identified such as Mobile/Web Application which in turn, from the GUI (*Graphical User Interface*) standpoint, can be subdivided in Basic/Advanced GUI objects, Task-based GUI objects, and so forth.

It is important to remark that an entity cannot be measured directly, but by means of its attributes. Attributes are quantified using metrics during the measurement process and are interpreted using indicators during the evaluation process. Figure 8 illustrates the link between Measurable Properties – attributes-, Measurement (light-blue box) and Evaluation (pink box). Therefore, Usability, a characteristic for the External Quality focus, combines a set of attributes for evaluating GUI objects. For the External Quality focus other characteristics (and attributes associated to other System sub-entities) such as Information Quality, Security, among others, can be used for ME Information Needs.

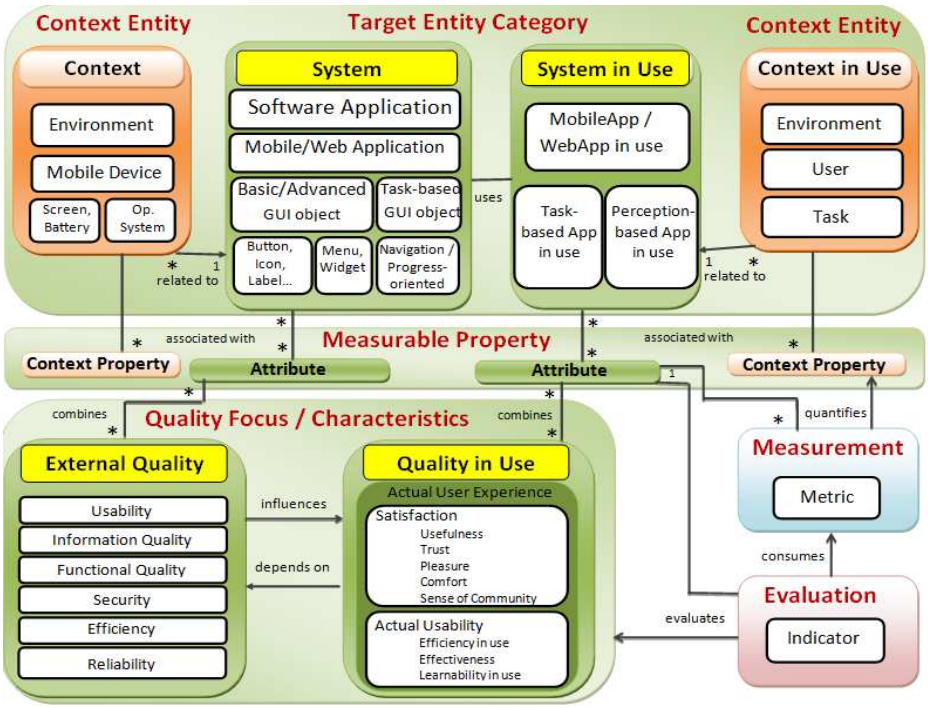

Figure 8: Rough schema derived from Figure 4, which relates the main building blocks for Target Entity (mainly related to GUI objects for System) and Context Entity Categories, Quality Focuses (just for External Quality and QinU), Measurable Properties, Measurement and Evaluation.

Figure 8 also depicts the System in Use box (i.e., the target entity super-category) with the corresponding Quality in Use box (i.e., the quality focus and the UX characteristic and sub-characteristics). Looking at the former box, two main sub-entity categories are identified viz., Task-based App in use, and Perception-based App in use. Concrete task-based app-in-use entities can be evaluated using attributes combined to Effectiveness, Efficiency in use and Learnability in use sub-characteristics (see definitions in Figure 7), which can be measured objectively. Conversely, perception-based app-in-use entities involve those subjective measures for Satisfaction sub-characteristics such as Usefulness, Trust, Pleasure, Comfort, etc.

Hence, Usability deals with the specification and evaluation of interface-based sub-characteristics and attributes of a system or product, while Actual Usability deals with the specification and evaluation of task-based sub-characteristics and attributes of an app in use, and Satisfaction with perception-based sub-characteristics and attributes.

Taking into account the ‘influences’ and ‘depends on’ relationships between quality views, Figure 8 shows that the QinU focus (System-in-Use Quality View) depends on the External Quality focus (System Quality View). Considering some empirical observations made in [26], we can indicate for instance that Actual Usability depends not only on Usability, but also on other sub-characteristics such as Information Quality and Efficiency.

Lastly, Figure 8 depicts two Context boxes, and Figure 4 represents *Context* as an Entity Category. Context is a special kind of entity category representing the state of the situation of a target entity to be

assessed, which is relevant for a particular ME Information Need. For instance, many works ([10, 21, 33], among others) argue that Context is paramount, as instantiation of QinU requirements must be done consistently in the same context so that evaluations and improvements can be accurately assessed and compared. Also context is somewhat important regarding the System Quality View. Specifically, System in Use is characterized by a Context-in-Use entity, which in turn can aggregate Environment, User and Task sub-entities (as suggested in Figure 1), while Context for System can be characterized by sub-entities such as Device, Screen, Operating System, etc. To describe the Context, *Context Properties* (Figure 4) are used which are also Attributes.

3.2 Integrated Strategies for Measurement, Evaluation and Improvement

As described in the Introduction Section, *integrated ME/MEC strategies* are the second pillar of our holistic quality evaluation and improvement approach. The fact of modeling quality views and their relationships is crucial for the aim of this pillar, since strategies are chosen considering quality views to be evaluated according to the ME/MEC project goal.

In this approach, an integrated strategy simultaneously supports three capabilities [37]: a *domain conceptual base and framework*, *process perspective specifications*, and *method specifications*. Regarding the first capability, Figure 4 shows the C-INCAMI conceptual base and framework. C-INCAMI explicitly specifies the ME/MEC terms, properties, relationships and constraints, in addition to their grouping into components. The second capability, the process specifications, usually describes a set of activities, tasks, inputs and outputs, pre- and post-conditions, artifacts, roles, and so forth. Additionally, process specifications can consider different process perspectives such as functional, behavioral, informational and organizational [11]. Usually, process specifications chiefly state what to do rather than indicate the particular methods and tools (resources) used by specific activity descriptions. The third capability provides the ability to specify methods, which ultimately represent the particular ways to perform the ME and MEC tasks.

So far, we have developed a couple of specific ME/MEC strategies, such as GOCAME and SIQinU. GOCAME is a strategy useful for ME project goals that embraces one quality view. That is, a project goal related to understand the current situation of an entity belonging to an entity category with regard to the corresponding quality focus.

On the other hand, SIQinU supports the QinU/External Quality/QinU evaluation and improvement cycles, starting evaluations from Task-based and/or Perception-based App-in-use entities, and their corresponding characteristics and attributes. So SIQinU embraces two quality views (System-in-Use Quality View and System Quality View) and explores the ‘influences’ and ‘depends on’ relationships between views.

Due to the experience gained in the last decade in the development and use of the above strategies [25, 26, 33, 36, 37], we have recently observed that different strategies can be applied to recurrent problems within given measurement, evaluation and change/improvement situations for specific projects' goals. Thus, we have envisioned to develop a set of strategy patterns that offer reusable and instantiable solutions. They are essentially “experience in a can” ready to be opened and used by evaluators. In the next section, we thoroughly specify two strategy patterns.

To summarize, our holistic quality approach can help software and web organizations to reach the

planning and performing of measurement, evaluation and change project goals in a systematic and disciplined way. The approach allows embracing different quality focuses and entity categories in a flexible yet structured manner by means of quality views with the final aim of defining and instantiating specific strategies to achieve ME/MEC project goals.

4 Strategy Patterns for Measurement, Evaluation and Improvement

In software and web engineering, a well-known definition for design patterns is "*each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice*" [15]. The idea was taken from Alexander [1] who was the first in introducing patterns for the urban domain, but the core concept was later applied to software and web domains. In other words, the idea of patterns is to provide a reusable and customizable solution to a recurrent problem in similar situations. Regarding this, we see different situations when establishing evaluation projects. Project goals can be aimed toward different evaluation purposes (e.g., understand, improve, predict, etc.), so a suitable strategy to achieve a given goal should be selected. Strategy patterns arise as a way for providing a solution in the instantiation of ME/MEC strategies considering the statement of the project goal, i.e., its purpose and the number of included qualities views.

We have specified a set of strategy patterns, following to some extent the pattern specification template used in [15]. 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)*: Project problem/goal solved by the strategy pattern; (5) *applicability*: Situations in which it can be applied; (6) *structure (solution)*: Generic structure and instantiable solution that the strategy pattern offers; (7) *known uses*: References of real usage; (8) *scenario of use*: Concrete example and illustration for the instantiated pattern.

It is worthy remarking that the ontology of quality views plays a central role in defining and instantiating strategy patterns. That is, without a clear specification of the terms and relationships for quality views, the further specification of strategy patterns could not be done appropriately. Specifically, the ontology of quality views fosters the specification and selection of appropriate strategy patterns and their instantiation regarding different ME/MEC project goals.

In the following sub-sections, we describe two strategy patterns: One involves one quality view and the other two quality views considering the 'influences' and 'depends on' relationships. Both are intended for improvement.

4.1 GoMEC_1QV: A Strategy Pattern for Improving Quality considering One Quality View

If a project has a goal such as 'Improve the Facebook mobileapp Usability', this means that it has an 'improve' purpose which embraces the *System Quality View*. This is because the concrete *Entity* is the 'Facebook mobile app' which belongs to the System *Entity Super-Category*, and the *Quality Focus* is External Quality. For this goal, it is necessary not only to understand the current situation of the entity at hand but also to perform changes on the entity, and then to re-evaluate it in order to gauge the improvement gain.

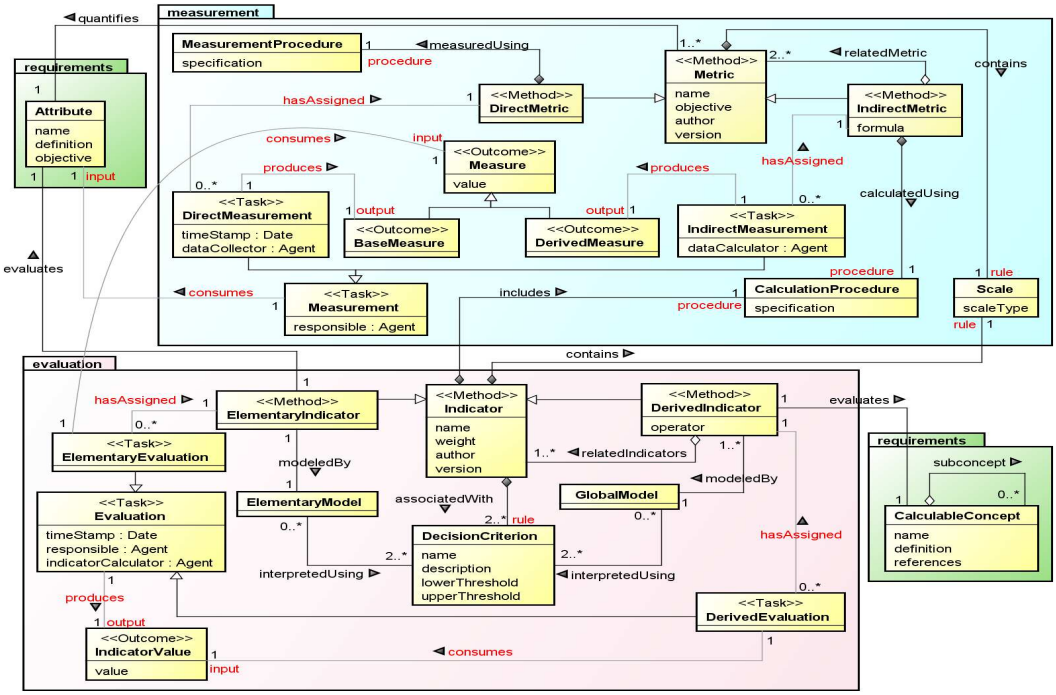

Figure 9: *Measurement* and *evaluation* components of the C-INCAMI conceptual framework enriched with process terms.

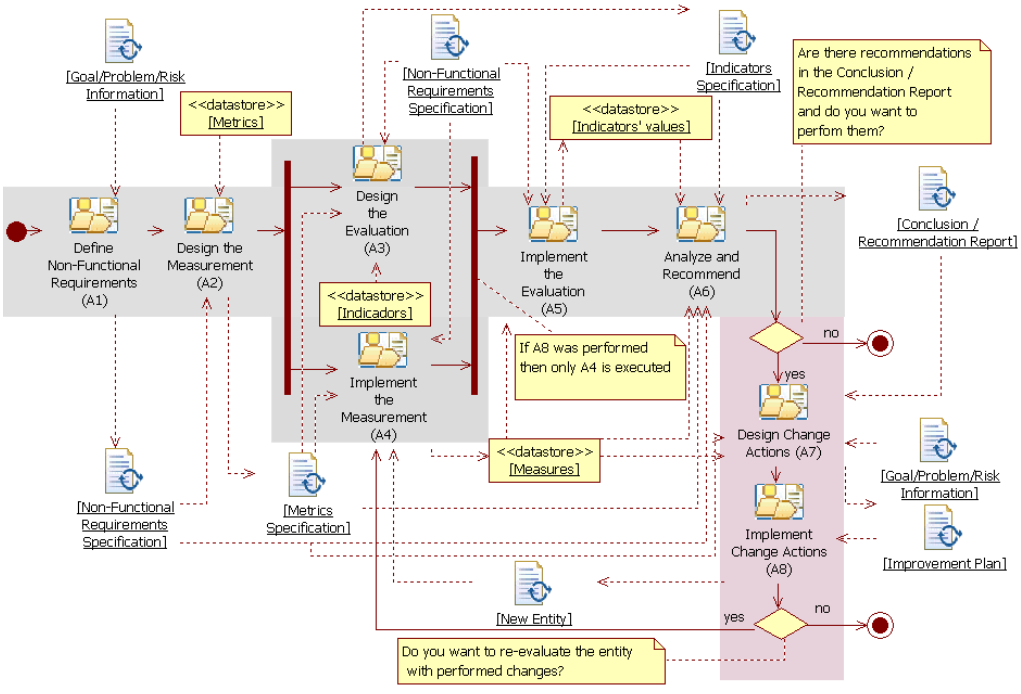

Figure 10: Generic process from the functional and behavioral perspectives for the GoMEC_1QV pattern.

Another project goal can have the same 'improve' purpose. However, the involved quality view could be a different one such as the Software Product Quality View or Resource Quality View. Hence, for a particular MEC project embracing one quality view, the same MEC strategy pattern should be selected and tailored accordingly. In short, for the intended quality view, the strategy pattern, i.e., its processes and methods should be instantiated appropriately. This pattern is the so-called *Goal-oriented Measurement, Evaluation and Change for One Quality View* (alias GoMEC_1QV). The items used to specify this pattern are the following:

Intent: To provide a solution in the instantiation of a measurement, evaluation, analysis and change strategy aimed at supporting a project improvement goal when one quality view is considered.

Motivation (Problem): The purpose is to understand the current situation of a concrete entity in a specific context for a set of characteristics and attributes related to a given quality focus and then change the entity and re-evaluate it in order to gauge the improvement gain, through the systematic use of measurement, evaluation, analysis and change activities and methods.

Applicability: This pattern is applicable in MEC projects where the purpose is to understand and improve the quality focus of the evaluated entity for one quality view, such as System, System-in-Use Quality Views, among others.

Structure (Solution): The pattern structure is based on the three capabilities of an integrated strategy viz., the specification of the conceptual framework for the MEC domain, the specification of MEC process perspectives, and the specification of MEC methods. GoMEC_1QV provides a generic course of action that indicates which activities should be instantiated during project planning. It also provides method specifications for indicating how the activities should be performed. Specific methods can be instantiated during scheduling and execution phases of the project. Below, we describe the structural aspects of the three strategy capabilities:

I. The concepts in the `non-functional requirements`, `context`, `measurement`, `evaluation`, `change`, and `analysis` components (Figures 4 and 9) are defined as sub-ontologies. The included terms, attributes and relationships belong to the MEC area. In Figure 9 we show just the main measurement and evaluation terms. Note that terms in the `measurement` and `evaluation` components are also enriched with terms from a generic process ontology [6] by means of stereotypes. These concepts are used consistently in the activities, artifacts, outcomes and methods of any ME/MEC strategy.

II. The process specification is made up from different perspectives, i.e., *functional* which includes activities, inputs, outputs, etc.; *behavioral*, which includes parallelisms, iterations, etc.; *organizational*, which deals with agents, roles and responsibilities; and *informational*, which includes the structure and interrelationships among artifacts produced or consumed by activities. Considering the functional and behavioral perspective, Figure 10 depicts the generic process for this pattern. The names of the eight (A1-A8) MEC activities must be customized taking into account the concrete quality view to be evaluated.

III. The method specification indicates how the descriptions of MEC activities must be performed. Tables 3 and 4 exemplify three method specification templates: one for a direct metric used as method specification for direct measurement tasks; one for an indirect metric, used in indirect measurements; and other for an elementary indicator, used in elementary evaluations. Note that

terms in method specification templates come from the ME conceptual base. Many other method specifications can be envisioned such as task usage log files, questionnaires, aggregation methods for derived evaluation, amongst others. For change activities traditional methods such as refactoring, re-structuring, re-parameterization, among others can be specified as well.

Known uses: GoMEC_1QV was used in a MEC project devoted to improve Usability and Information Quality attributes of a shopping cart, i.e., from the System Quality View through refactoring as change method [36]. Besides, this pattern was instantiated in a MEC project for the Resource Quality View [37].

Scenario of use: The present example stems from the Facebook's mobileapp (v3.8 for Android) Usability case study performed in Dec. 2013 [33], and then replicated for the v14 in Sept. 2014, applying also the same Usability requirements.

Table 3: Method specification templates for the Measurement task: a) for a Direct Metric; b) for an Indirect Metric.

a) **Direct Metric Template**

Quantified Attribute name:
Metric name:

Objective:
Author: **Version:**

Measurement Procedure:
Type:
Specification:

Scale: [Numerical | Categorical]
Scale Type name:
Value Type: Representation:

Unit:
Name:
Description: Acronym:

Tool:
(Note: Information about the used tool if any)

b) **Indirect Metric Template**

Quantified Attribute name:
Metric name:

Objective:
Author: **Version:**

Calculation Procedure:
Procedure specification:
Formula:

Scale: [Numerical | Categorical]
Scale Type name:
Value Type: Representation:

Unit:
Name:
Description: Acronym:

Tool:
(Note: Information about the used tool if any)

Related Direct Metrics:

Table 4: Method specification template for the Elementary Evaluation task: Elementary Indicator.

Elementary Indicator Template
Interpreted Attribute name: **Indicator name:** **Author:** **Version:**
Elementary Model: Elementary Model specification: Decision Criteria [Acceptability Levels] Name: Range: Description:
Scale: [Numerical | Categorical] Scale Type name: Value Type: Representation:
Unit Name: Description: Acronym:

The project's goal is to improve the entity by evaluating, analyzing and detecting Usability problems, and recommending change actions for fixing weaknesses. So the Quality View is System Quality View, i.e., the Entity Super-Category is System and the concrete Entity is Facebook mobile app. The Quality Focus is External Quality (EQ) where the evaluated characteristic is Usability and its related sub-characteristics as Understandability, Learnability, Operability, User Error Protection and UI Aesthetic.

For the above project goal, the GoMEC_1QV strategy pattern should be selected and instantiated for the above-mentioned quality view. Therefore, the eight generic activities that the pattern provides as a solution (Figure 10) are instantiated. For example, A1 is now renamed 'Define Non-Functional Requirements for EQ', A2 is renamed as 'Design Measurement for EQ', and so on. In the rest of this sub-section, the eight instantiated activities are illustrated.

(A1) Define Non-Functional Requirements for EQ: The instantiated A1 activity produces the 'Non-Functional Requirements Specification for EQ' document, which includes the 'Information Need Specification for EQ' (Figure 11), and the 'Requirements Tree for EQ' artifacts. In Table 5, the Usability sub-characteristics and attributes definitions included in the 'Requirements Tree for EQ' for this case study are shown.

ME Information Need's purpose: Improve **User Viewpoint:** Final user
Quality Focus: External Quality **Entity Super-Category:** System
Entity Category: Social Network Application **Entity:** Facebook mobile app, v.14 for Android

Context Properties: *Mobile device type*: 'Mobilephone', *Screen size*: '540x960px/4.3inches', *Mobilephone generation*: 'Full-sized smartphone', amongst others.

Calculable Concepts: *Usability* and its related sub-characteristics: *Understandability*, *Learnability*, *Operability*, *User Error Protection*, and *User Interface Aesthetics*

Figure 11: Summarized Information Need artifact produced in the instantiated A1 activity.

Table 5: Definition of Usability sub-characteristics and attributes –in *italic*.

Characteristic/*Attribute*	**Definitions in the 2Q2U v2.0 quality model**
1 Usability	See the definition in Figure 6.
1.1 Understandability	See the definition in Figure 6.
1.1.1 Familiarity	Degree to which the user understand what the application, system's functions or tasks are about, and their functionality almost instantly, mainly from initial impressions
1.1.1.1 Global organization scheme understandability	Degree to which the application scheme or layout is consistent and adheres to either de facto or industry standard to enable users to instantly understand its function and content.
1.1.1.2 Control icon ease to be recognized	Degree to which the representation of the control icon follows or adheres to an international standard or agreed convention.
1.1.1.2.1 Main control icon ease to be recognized	Degree to which the representation of the main controls icons follows or adheres to an international standard or agreed convention.
1.1.1.2.2 Contextual control icon ease to be recognized	Degree to which the representation of the contextual controls icons follows or adheres to an international standard or agreed convention.
1.1.1.3 Foreign language support	Degree to which the application functions, controls and content has multi-language support enabling user to change his/her language of preference.
1.2 Learnability	The definition was given in Figure 6.
1.2.1 Feedback Suitability	Degree to which mechanisms and information regarding the success, failure or awareness of actions is provided to users to help them interact with the application.
1.2.1.1 Current location feedback appropriateness	Degree to which users are made aware of where they are at the current location by an appropriate mechanism.

1.2.1.2 Alert notification feedback appropriateness	Degree to which users are made aware of new triggered alerts that they are involved by an appropriate mechanism.
1.2.1.3 Error message appropriateness	Degree to which meaningful error messages are provided upon invalid operation so that users know what they did wrong, what information was missing, or what other options are available.
1.2.2 Helpfulness	Degree to which the software product provides help that is easy to find, comprehensive and effective when users need assistance
1.2.2.1 Context-sensitive help appropriateness	Degree to which the application provides context sensitive help depending on the user profile and goal, and current interaction.
1.2.2.2 First-time visitor help appropriateness	Degree to which the application provides an appropriate mechanism (e.g. a guided tour, etc) to help beginner users to understand the main tasks that they can do.
1.3 Operability	See the definition in Figure 6.
1.3.1 Data Entry Ease	Degree to which mechanisms are provided which make entering data as easy and as accurate as possible.
1.3.1.1 Defaults	Degree to which the application provides support for default data.
1.3.1.2 Mandatory entry	Degree to which the application provides support for mandatory data entry.
1.3.1.3 Widget entry appropriateness	Degree to which the application provides the appropriate type of entry mechanism in order to reduce the effort required.
1.3.2 Visibility (synonym Optical Legibility)	Degree to which the application enables ease of operation through controls and text that can be seen and discerned by the user in order to take appropriate actions.
1.3.2.1 Color visibility appropriateness	Degree to which the main GUI object (e.g. text, control, etc.) color compared to the background color provides sufficient contrast and ultimately appropriate visibility.
1.3.2.1.1 Brightness difference appropriateness	Degree to which the foreground color of the GUI object (e.g. text, control, etc.) compared to the background color provides appropriate brightness difference.
1.3.2.1.2 Color difference appropriateness	Degree to which the foreground text or control color compared to the background color provides appropriate color difference.
1.3.2.2 GUI object size appropriateness	Degree to which the size of GUI objects (e.g. text, buttons, and controls in general) is appropriate in order to enable users to easily identify and operate them.
1.3.2.2.1 Control (widget) size appropriateness	Degree to which the size of GUI controls is appropriate in order to enable users to easily identify and operate them.
1.3.2.2.2 Text size appropriateness	Degree to which text sizes and font types are appropriate to enable users to easily determine and understand their meaning.
1.3.3 Consistency	Degree to which users can operate the task controls and actions in a consistent and coherent way even in different contexts and platforms.
1.3.3.1 Permanence of controls	Degree to which main and contextual controls are consistently available for users in all appropriate screens or pages.
1.3.3.1.1 Permanence of main controls	Degree to which main controls are consistently available for users in all appropriate screens or pages.
1.3.3.1.2 Permanence of contextual controls	Degree to which contextual controls are consistently available for users in all appropriate screens or pages.
1.3.3.2 Stability of controls	Degree to which main controls are in the same location (placement) and order in all appropriate screens.
1.4 User Error Protection	See the definition in Figure 6.
1.4.1 Error Management	Degree to which users can avoid and recover from errors easily.
1.4.1.1 Error prevention	Degree to which mechanisms are provided to prevent mistakes.
1.4.1.2 Error recovery	Degree to which the application provides support for error recovery.
1.5 UI Aesthetics	See the definition in Figure 6.
1.5.1 UI Style Uniformity	Degree to which the UI provides consistency in style and meaning.
1.5.1.1 Text color style uniformity	Degree to which text colors are used consistently throughout the UI with the same meaning and purpose.
1.5.1.2 Aesthetic harmony	Degree to which the UI shows and maintains an aesthetic harmony regarding the usage and combination of colors, texts, images, controls and layouts throughout the whole application.

(A2) Design the Measurement for EQ: In the MEC project's scheduling phase, metrics (as methods) are assigned to the instantiated activities. During the A2 activity, metrics are selected and assigned to quantify all attributes of the requirements tree.

Metrics are retrieved from a repository (see Metrics <<datastore>> in Figure 10) and their

specifications are based on templates such as the one shown in Table 3. For instance, the *Ratio of Main Controls Permanence* (%MCP) Indirect Metric quantifies the *Permanence of main controls* attribute (coded as 1.3.3.1.1 in Table 5). The metric's objective is "*to determine the percentage of permanence for controls from the set of main controls (buttons) in the application selected screens*". Figure 12 shows full details of the metric specification.

Attribute: Name: *Permanence of main controls*.
Indirect Metric: Name: *Ratio of Main Controls Permanence* (%MCP);
Objective: To determine the percentage of permanence for controls from the set of main controls in the application selected screens
Author: Santos L.; **Version**: 1.0
Calculation Procedure: Formula: $\%\mathrm{MCP} = \left[\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\mathrm{MCPL}_{ij}}{\mathrm{m*n}}\right] * 100$; where m is the number of application main controls and n is the number of application selected screens; with m, n > 0
Numerical Scale:
Representation: Continuous; **Value Type**: Real; **Scale Type name**: Ratio
Unit Name: Percentage; **Acronym**: %
Related Metrics: Main control permanence level (MCPL)

Related Direct Metric: Name: *Main Control Permanence Level* (MCPL);
Objective: To determine the permanence level of a selected control in a given application screen;
Author: Santos L.; **Version**: 1.0
Measurement Procedure: Type: Objective;
Specification: The expert inspects the main controls bar in a given screen in order to determine whether the button is available or not, using the 0 or 1 allowed values. Where 0 means the main button is absent in the screen, and 1 means the main button is present in the screen
Numerical Scale:
Representation: Discrete; **Value Type**: Integer; **Scale Type**: Absolute
Unit Name: Control

Figure 12: Indirect and direct metric specifications for the *Permanence of main controls* attribute.

(A3) Design the Evaluation for EQ: This activity consists of defining elementary indicators to map measures values to a new numeric or categorical value in order to interpret the measured value. Also derived indicators are defined, which allow interpreting high-level abstraction requirements, that is, to interpret characteristics and sub-characteristics by means of a global model. Figure 13 shows the elementary indicator specification for the attribute *Permanence of main controls*, following the elementary indicator specification template shown in Table 4.

Attribute: *Permanence of main controls*
Elementary Indicator: Name: *Performance Level of the Permanence of Main Controls (P_MCP)*
Author: Santos L.; **Version**: 1.0

Elementary Model:
Specification: the mapping is: P_MCP = %MCP.

Decision Criterion: [Three Acceptability Levels]
Name 1: Unsatisfactory **Range**: if 0 ≤ P_MCP < 60 Description: Indicates change actions must be taken with high priority.
Name 2: Marginal **Range**: if 60 ≤ P_MCP < 80 Description: Indicates a need for improvement actions.
Name 3: Satisfactory **Range**: if 80 ≤ P_MCP ≤ 100 Description: Indicates no need for current actions.

Numerical Scale: **Value Type**: Real; **Scale Type**: Ratio
Unit/Name: Percentage; **Acronym**: %

Figure 13: Elementary Indicator specification for the *Permanence of main controls* attribute.

Note that acceptability levels and ranges can be designed and better calibrated if historical data or specific benchmarks exist in the company or elsewhere.

(A4) Implement the Measurement for EQ: The pattern's generic process establishes A4 as the next activity to be instantiated, though A3 and A4 can be executed in parallel , as shown in Figure 10. Hence, 'Implement the Measurement for EQ' produces the measures for each attribute.

For example, using the above Indirect Metric specification for quantifying the 1.3.3.1.1 attribute, A4 allows recording its availability per each 'Main button' of the 'Main controls bar' for each Facebook screen in which the button must remain permanent. So evaluators can easily understand what concrete button is absent in each screen for a further change action, if necessary. The final calculated value for this indirect metric was 46.2%. Additionally, all intermediate values for related direct metrics were also recorded in the Measures data store.

(A5) Implement the Evaluation for EQ: From the measures obtained in A4 and using the selected elementary indicators and derived indicators designed in A3, the A5 activity (renamed as 'Implement the Evaluation for EQ') must be instantiated and executed. This activity produces the Elementary and Derived Indicators' values shown in 2nd and 3rd columns of Table 6, for the evaluation performed in Sept. 2014.

Notice that in Table 6, the metaphor of the three-colored semaphore is used to identify the acceptability level of satisfaction achieved by each attribute/sub-characteristic. For example, the red-colored semaphore (with values within the 0-60 range, in the percentage scale) indicates an 'Unsatisfactory' acceptability level. This means that change actions must be done urgently.

Table 6: *Requirements Tree for EQ* with indicator values yielded in the 2014 case study. (Note: EI means Elementary Indicator; DI means Derived Indicator).

Characteristic / Sub-characteristic / *Attribute*	**EI**	**DI**
1 Usability		**65.1**🟡
1.1 Understandability		**89.9**🟢
1.1.1 Familiarity		89.9🟢
1.1.1.1 Global organization scheme understandability	100🟢	
1.1.1.2 Control icon ease to be recognized		74.8🟡
1.1.1.2.1 Main control icon ease to be recognized	71.4🟡	
1.1.1.2.2 Contextual control icon ease to be recognized	88.9🟢	
1.1.1.3 Foreign language support	100🟢	
1.2 Learnability		**66.4**🟡
1.2.1 Feedback Suitability		85.6🟢
1.2.1.1 Current location feedback appropriateness	80.3🟢	
1.2.1.2 Alert notification feedback appropriateness	100🟢	
1.2.1.3 Error message appropriateness	75🟡	
1.2.2 Helpfulness		49.1🟡
1.2.2.1 Context-sensitive help appropriateness	54.5🔴	
1.2.2.2 First-time visitor help appropriateness	0🔴	
1.3 Operability		**80.4**🟢
1.3.1 Data Entry Ease		90🟢
1.3.1.1 Defaults	100🟢	
1.3.1.2 Mandatory entry	50🔴	
1.3.1.3 Widget appropriateness	100🟢	
1.3.2 Visibility (synonym **Optical Legibility**)		81.5🟢

1.3.2.1 Color visibility appropriateness		100 🟢
1.3.2.1.1 Brightness difference appropriateness	100 🟢	
1.3.2.1.2 Color difference appropriateness	100 🟢	
1.3.2.2 GUI object size appropriateness		63 🟡
1.3.2.2.1 Control (widget) size appropriateness	100 🟢	
1.3.2.2.2 Text size appropriateness	42.1 🔴	
1.3.3 Consistency		74.6 🟡
1.3.3.1 Permanence of controls		55.7 🔴
1.3.3.1.1 Permanence of main controls	46.2 🔴	
1.3.3.1.2 Permanence of contextual controls	100 🟢	
1.3.3.2 Stability of controls	95.5 🟢	
1.4 User Error Protection		**8.4** 🔴
1.4.1 Error Management		8.4 🔴
1.4.1.1 Error prevention	0 🔴	
1.4.1.2 Error recovery	16.7 🔴	
1.5 UI Aesthetics		**91.1** 🟢
1.5.1 UI Style Uniformity		91.1 🟢
1.5.1.1 Text color style uniformity	95 🟢	
1.5.1.2 Aesthetic harmony	89.5 🟢	

(A6) Analyze and Recommend for EQ: After finishing ME activities, the instantiated A6 activity should be performed. Table 7 shows a fragment of the A6 generated document named 'Recommendation Report for EQ'. This report contains one or more recommendations for attributes that did not meet the 'Satisfactory' level, i.e., for those with red- or yellow-colored semaphores. Each recommendation has also a priority. E.g., in Table 7, 'H' means high priority. Recommendations enable to design and perform change actions for improvement.

Looking at the elementary indicator values in Table 6, seven attributes fell in the 'Unsatisfactory' acceptability level, namely: 'Context-sensitive help appropriateness' (1.2.2.1), 'First-time visitor help appropriateness' (1.2.2.2), 'Mandatory entry' (1.3.1.2), 'Text size appropriateness' (1.3.2.2.2), 'Permanence of main controls' (1.3.3.1.1), 'Error prevention'(1.4.1.1) and 'Error recovery' (1.4.1.2). So, at least, seven recommendations must be done with high priority for designing change actions.

For instance, for the 'Permanence of main controls' attribute, the R2.1 recommendation in Table 7 establishes "*ensure that in the set of selected screens, the main controls bar has always the same main buttons*" and that should be pursued as a high priority change action. Figure 14 shows two selected screenshots belonging to the set of appropriate screens that were evaluated (34 out of 38) for the Facebook mobile app, version 14 for Android. The 'Main controls bar' sub-entity has seven 'Main buttons'. Also, it can be observed in the right screen that the specific "Chat button" is missing.

Table 7: Fragment of the *Recommendation Report for EQ* artifact generated in the instantiated A6 activity. (Note: H means High priority, i.e., an urgent action is recommended).

	Recommendation (R)	**Attribute**	**Priority**
R1	**1.** To ensure that in the set of selected screens, those mandatory form fields have the suitable support for preventing missing data entry.	*Mandatory entry* (1.3.1.2)	H
R2	**1.** To ensure that in the set of selected screens, the main controls bar has always the same main buttons.	*Permanence of main controls* (1.3.3.1.1)	H

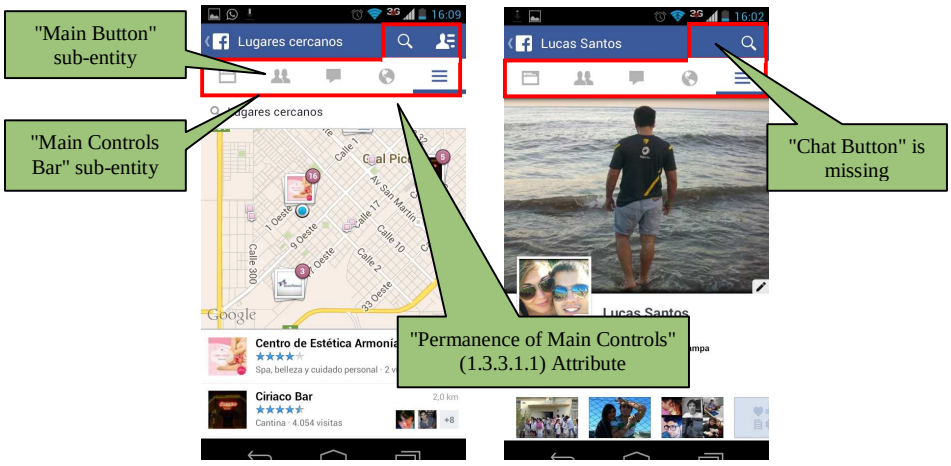

Figure 14: Two Facebook v14 screenshots. Left one shows the 'Main Controls Bar' sub-entity, which is composed of seven 'Main Buttons'. Right one highlights the missing 'Chat button'.

(A7) Design Change Actions for EQ: The next A7 activity specified in GoMEC_1QV should be instantiated. Thus, A7 is renamed now as 'Design Change Actions for EQ'. Table 8 shows a fragment of the 'Improvement Plan for EQ' artifact. Basically, per each recommendation (R) in Table 7, the planned change actions (CA), the CA source for each attribute (Measures repository), and the method type to be used for the change action are described.

For the 1.3.3.1.1 attribute, the change action is CA2.1 in Table 8, derived from R2.1 in Table 7. It indicates: "*add those missing main buttons in the main controls bar per each screen with the problem detected*". To this end, the 'GUI refactoring' change method can be used (as in [36]). Besides, in the measurement registry for 1.3.3.1.1 (Measures <<datastore>>) can be found the corresponding screen ID where each concrete button is missing. Therefore, we can affirm that a good metric specification can help in planning and performing change actions.

(A8) Implement Change Actions for EQ: The last activity is devoted to implement change actions planned in the A7 activity. It should be noted that since Facebook is a proprietary system, changes actually couldn't be performed since we didn't have access to its source code and GUI objects.

Once changes are made through the instantiation of A8, the GoMEC_1QV course of action establishes that the new system version should be re-evaluated. To this aim, A4, A5 and A6 must be performed again, as specified in Figure 10. So the achieved improvement gain can be compared with the previous version. Thus, once the CA2.1 change action is performed on the app, the elementary indicator value for 'Permanence of main controls' will rise from 46.2% (red) to 100% (green). If all recommended changes for weakly benchmarked indicators were performed, the overall level of satisfaction for Usability could reach 100% for the target entity.

Table 8: Fragment of the *Improvement Plan for EQ* produced in the instantiated A7 activity.

	Change Action (CA)	**CA sources**	**Method**
CA1	**1.** Provide a function on the form mandatory field/control for checking missing data entry. **2.** Add a suitable visual indicator (e.g., "*") to each form's mandatory field/control.	Use the Measures/Measurement registry for 1.3.1.2 to find the form field/control ID with the problem.	Programming GUI refactoring
CA2	**1.** Add those missing main buttons in the main controls bar per each screen with the problem detected.	Use the Measures/Measurement registry for 1.3.3.1.1 to find the screen/main button ID with the problem.	GUI refactoring

4.2 GoMEC_2QV: A Strategy Pattern for Improving Quality considering Two Quality Views

When the project involves an improvement/change goal for two quality views, the *Goal-oriented Measurement, Evaluation and Change for Two Quality Views* strategy pattern (alias GoMEC_2QV) should be chosen. This strategy pattern addresses the fact that one quality view can be improved from evaluation-driven change actions taken on other quality view thanks to the aforementioned 'influences' and 'depends on' relationships between quality views.

As highlighted in Figure 3, the System Quality View influences the System-in-Use Quality View. Hence, by evaluating and improving the External Quality focus of a System is a means for improving the QinU focus of the System in Use. Conversely, evaluating the QinU can provide feedback to improve the External Quality by exploring the 'depends on' relationship. Note that the GoMEC_2QV strategy pattern can be instantiated with other two related quality views, such as Resource and Process Quality views, taking into account for instance that a resource quality (e.g. a new integrated tool) influences the process quality (e.g. a development process), and vice versa the process quality depends on the resource quality. The pattern specification is described below:

Intent: To provide a solution in the instantiation of a measurement, evaluation and improvement strategy aimed at supporting a project improvement goal when two related quality views are considered.

Motivation (Problem): Considering that a given quality view depends (directly or indirectly) on other quality view (regarding the 'dependent view' and 'independent view' roles as specified in Figure 4), the purpose is to improve the current situation of a concrete entity belonging to a dependent quality view by applying evaluation-driven changes to other entity belonging to a related independent quality view.

Applicability: This pattern is applicable in MEC projects where the purpose is to improve the quality focus of an entity belonging to a dependent view by means of improving the quality focus of an entity by performing changes on it that belongs to an independent view. For example, improving a system-in-use app (from the System-in-Use Quality View standpoint) by means of improving the system application (from the System Quality View standpoint). The 'depends on' relation can also be considered in a transitive way, e.g., improving the System Quality View by means of improving the Resource Quality View.

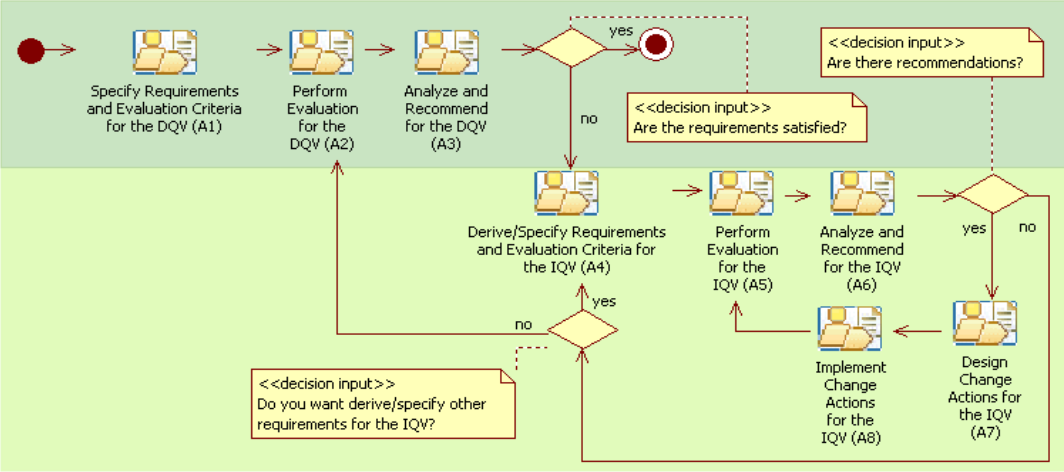

Figure 15: Generic process for the GoMEC_2QV pattern regarding functional and behavioral process perspectives. (Note: DQV means Dependent Quality View; IQV means Independent Quality View)

Structure (Solution): The pattern structure is based on the three capabilities of an integrated strategy. The specification of the conceptual framework for the MEC domain, and the specification of MEC methods are reused from the GoMEC_1QV pattern, so you can refer to (I) and (III) structure items, in sub-section 4.1. But for the process specification, GoMEC_2QV provides a generic course-of-action solution which indicates what activities must be instantiated during the project planning. Figure 15 depicts the generic process for this pattern. The names of the MEC activities must be tailored taking into account the concrete quality focuses to be evaluated.

Known uses: GoMEC_2QV was used to instantiate the so-called SIQinU strategy which was documented in an industrial case study presented in [26].

Scenario of use: An instantiation example for this strategy pattern is the case study conducted in mid-2010 in a real software testing enterprise with headquarters in Beijing, which examined JIRA, a commercial software defect tracking system. The whole case study was documented in [26], so in this item, we will only highlight aspects of the general process, exemplifying the use of the dependent and independent quality views.

Summarizing the study in [26], the project's goal was to understand and improve the current situation of the JIRA system in use (as concrete entity) from the beginner user viewpoint performing the most frequently used *Entering a new defect* task regarding Actual Usability (see its definition in Figure 7). The improvement was accomplished by means of deriving and understanding related External Quality attributes and performing changes on the JIRA system for those weakly benchmarked External Quality indicators. So this process embraces two quality views.

On one hand, the System-in-Use Quality View with a quality focus on QinU (with Actual Usability characteristic linked to this focus) for the System in Use entity super-category, particularly, for the task-based JIRA in use concrete sub-entity named Entering a new defect. (Recall Figure 8, where two sub-entity categories are depicted: Task-based App in use, and Perception-based App in use). On the other hand, the System Quality View with a quality focus on External Quality (with Usability and Information Quality characteristics linked to this focus) for the System entity super-

category.

Therefore, since two quality views intervened in the project's goal, the GoMEC_2QV strategy pattern was selected as the most suitable for instantiation. Figure 15 shows the generic process for GoMEC_2QV. Following this process, we summarize some aspects of the project goal and the pattern instantiation.

The goal was to evaluate JIRA from both QinU and External Quality viewpoints with the ultimate objective of making improvements on the QinU task performance. So, starting with QinU –the dependent quality view-, the specific task (Entering a new defect) and context of use were specified (A1 activity in Figure 15). It was necessary to collect user behavior data from log files that were derived through, for example, adding snippets of code in JIRA that allowed recording usage data. Next, the Perform the Evaluation for QinU (A2) activity was carried out with the aim of getting measures and indicators values for Actual Usability, specifically for Effectiveness, Efficiency and Learnability in use. This allowed us to understand the QinU acceptability levels met for their indicators. Then, by performing a preliminary analysis (A3 activity) we found that some Actual Usability attributes were not satisfied. This enable us to derive External Quality attributes (the independent quality view) that can have effect (influence) on QinU. So, after A4 we did the evaluation for External Quality requirements (A5 activity) and then, we analyzed and proposed recommendations for improvements (A6 activity) regarding weakly benchmarked External Quality indicators.

After having designed and performed the External Quality changes (A7 and A8 activities) on the JIRA web app using re-parameterization as change method, the new JIRA version was re-evaluated from the External Quality point of view. As there was no more improvement recommendations, we re-evaluated QinU performing A2 again. This re-evaluation was achieved by conducting user testing with the same user group in the same daily environment (context) with the new JIRA version. Therefore, the QinU improvement gain (A3 activity) was determined.

Ultimately, we were able to determine how the dependent quality view was enhanced by improvements made in the independent quality view.

5 Discussion about other Strategy Patterns

As we have commented above, a strategy pattern is a way of packaging general and reusable solutions for common and recurrent measurement, evaluation and change/improvement problems or situations for specific projects' goals. Hence, according to the project goal and the amount of involved quality views a strategy pattern should be selected and retrieved from a catalogue of strategy patterns. Each pattern stored in the catalogue should be compliant with the strategy pattern specification shown in Section 4.

In the previous section we have analyzed the GoMEC_1QV strategy pattern which is applied when the project goal states that it is necessary not only to understand the current situation of the entity at hand but also to perform changes on it, re-evaluate it, and gauge the improvement gain achieved. This pattern is instantiated for a MEC project goal considering just one quality view. We have illustrated this pattern in a case study for improving the Usability of the Facebook's social network mobile app. Usability was linked in this case study to the External Quality focus. However, we may also instantiate this pattern to improve, for instance, the Usability regarding the Internal Quality focus. The sub-entity category for the Internal Quality focus can be an artifact at early stages of development, such as an

architectural design, as presented in [14].

Another strategy pattern described in the previous section was the GoMEC_2QV, which gives a solution for an improvement project goal that involves two quality views and their 'influences' and 'depends on' relationships. This relations imply that one quality view plays the role of *dependent quality view*, while the other plays the role of *independent quality view*. For example, if we consider the System Quality View and the System-in-Use Quality View, these relations embrace the hypothesis [21] that evaluating and improving the External Quality focus of a system is one means for improving the QinU focus of a system in use. Additionally, understanding the QinU problems may provide feedback for deriving External Quality attributes that if improved could impact positively in the system quality. Furthermore, we can envision valid and interesting relationships for instance between Resource Quality View and Product Quality View. That is, by evaluating and improving the resource quality can be one means for improving the Internal Quality focus of a product. E.g., changes in the development team can impact positively in the architectural design.

On the other hand, GoME_1QV is the alias of the strategy pattern used to provide a solution in the instantiation of a ME strategy aimed at supporting just an understanding goal when one quality view is considered. This strategy pattern should be selected when the project goal is just to understand the current situation of an entity with regard to the corresponding quality focus. The generic process of GoME_1QV consists of six activities, which are the gray-colored A1-A6 activities in Figure 10. This is the simplest pattern to be instantiated. So far, it is also the mostly used in the ME projects we have performed, e.g., in the evaluation of a mashup application [34], a shopping cart [35], among others. This pattern can be used to evaluate not only any quality focus depicted in Figure 3 but also to other quality focuses such as Service Quality, amongst others.

Regarding the amount of views, a strategy pattern where three quality views intervene can be instantiated as well. For instance, we can mention GoMEC_3QV where the three yellow-colored Software Product, System and System-in-Use Quality Views in Figure 3 can be evaluated. Both Usability (for Internal Quality and External Quality) and User Expierence (for QinU) can be linked to the three quality views accordingly. In this sense, a project goal can be "to improve the quality in use of the Facebook mobile application by means of improving the system and software product quality". The GoMEC_3QV specification is not shown in this paper due to brevity reasons. However, the reader can surmise that it implies changes on two views.

So far, we have analyzed ME/MEC strategy patterns for providing solutions in the instantiation of strategies for specific project goals. Nevertheless, it is also possible to have a project goal related to the evaluation, comparison and selection of competitive entities such as selecting the best alternative system or product considering quality and/or cost performance indicators. In this direction, CMMI [9] establishes the *Decision Analysis and Resolution* process area aimed at evaluating, analyzing and performing decision making for identified alternatives (competitive entities) against established evaluation criteria using a formal evaluation process. So, we envision specifying a new pattern for this well-known situation. The intention of this strategy pattern is to provide a solution in the instantiation of a strategy that allows evaluating and comparing competitive entities for selecting one regarding the established quality/cost requirements and decision criteria.

Finally, it is worthwhile to remark that the inclusion of strategy patterns in our holistic quality evaluation and improvement approach makes it scalable. Thanks to the conception of different strategy

patterns applied to related quality views, our approach is scalable since the quality multi-view modeling framework (discussed in sub-section 3.1) supports many quality views and their relationships that ME/MEC project goals may deal with. Also the framework can be extended to specify not only quality but also cost requirements, since the *Quality View* and the *Cost View* are both *Calculable-Concept Views* as represented in Figure 2.

6 Conclusions and Future Work

In this work, we have enhanced our holistic quality evaluation and improvement approach by strengthening the two pillars of its architecture. First, for the quality multi-view modeling framework, we have defined the ontology of quality views aimed at adding conceptual robustness to our approach in addition to the ability to support semantic processability. In the process of building this ontology, we have reviewed related literature about quality views and multi-view modeling frameworks. Specifically, in Section 2 we have discussed that an ontology, taxonomy or glossary of terms for this domain does not exist based on the research we have done. However, the works of Garvin, Morasca, ISO 9126-1 and 25010, among others, were valuable sources of consultation.

Second, due to the lessons learnt in the development of concrete ME/MEC strategies during the last decade, we have envisioned the opportunity to generalize and distill the gained knowledge into strategy patterns. The benefits of having documented patterns are well known. Hence, we have contributed in the specification of a set of strategy patterns to be applied in the domain of ME/MEC projects. Also, we have discussed why the modeling of quality views and their 'influences' and 'depends on' relationships, in conjunction with ME/MEC project goals are key aspects to defining strategy patterns. Finally, we have documented two strategy patterns both for the same improvement goal but considering one and two quality views respectively. We have illustrated the instantiation of the GoMEC_1QV strategy pattern for the Facebook's mobileapp Usability case study, and we have also provided a description of the GoMEC_2QV instantiation considering Actual Usability evaluation and improvement.

As ongoing research, we are currently developing two ontologies whose terms are grouped in the `business_goal` and `project` components. Additionally, the ME Information Need in the `non-functional requirements` component (see Figure 4) is linked with the Business Goal term. A business goal can be formulated at different organizational levels, such as operational, tactical and strategic. The `project` component includes the Project term which operationalizes goals, both business and information need goals. Moreover this component includes the Strategy term, which helps to achieve goals. We envision to extend our holistic quality evaluation and improvement approach to give support in achieving multi-level goals following to some extent the GQM$^+$Strategy approach [2].

Considering the semantic processability, we also envision the development of a strategy pattern recommender system as a practical use of the quality views ontology in the context of the holistic approach. This recommender system can be useful when an organization establishes ME/MEC project goals. 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 the goal. This system could also be extended for multi-level goals.

Acknowledgements

We thank the support given by Science and Technology Agency of Argentina, in the PICT 2014-1224 project at Universidad Nacional de La Pampa. Also, Belen Rivera thanks the support given by the co-funded CONICET and UNLPam grant.

References

1. Alexander C: The Timeless Way of Building. Oxford University Press, (1979)
2. Basili V., Lindvall M., Regardie M., Seaman C., Heidrich J., Jurgen M., Rombach D.,Trendowicz A.: Linking Software Development and Business Strategy through Measurement. IEEE Computer, 43(4), pp. 57–65, (2010)
3. Basili V., Caldiera G., Rombach H. D.: The Goal Question Metric Approach. In: Encyclopedia of Software Engineering, (1), pp 528-532, (1994)
4. Basili V.: Software Development: A Paradigm for the Future. Proceedings of the 13th Annual International Computer Software & Applications Conference (COMPSAC), Keynote Address, Orlando, FL, (1989)
5. Baskerville R., Levine L., Pries-Heje J., Ramesh B., Slaughter S.: How Internet Companies Negotiate Quality, IEEE Computer 34(5), pp. 51-57, (2001)
6. 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)
7. Bevan N.: Extending Quality in Use to provide a Framework for Usability Measurement. LNCS 5619, Springer, HCI Int'l 2009, San Diego, USA, pp. 13-22, (2009)
8. Briand L., Morasca S., Basili V.: Property-based Software Engineering Measurement. IEEE Trans. on Software Engineering, Vol. 22, pp.68-86, (1996)
9. CMMI Product Team. CMMI for Development, Version 1.3 (CMU/SEI-2010-TR-033). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, (2010)
10. Crowley J., Coutaz J., Rey G., Reignier P.: Perceptual Components for Context Aware Computing. 4th Int'l Conference on Ubiquitous Computing (UbiComp '02), Borriello G. and Holmquist L. (Eds.), Springer-Verlag, London, UK, pp. 117-134, (2002)
11. Curtis B., Kellner M., Over J.: Process Modelling. Com. of ACM, 35:(9), pp.75-90, (1992)
12. Fernández-López M., Gómez-Pérez A., Juristo N.: METHONTOLOGY: From Ontological Art Towards Ontological Engineering. Spring Symposium on Ontological Engineering of AAAI, Stanford University, California, pp. 33-40, (1997)
13. Folmer E., Bosch J.: Experiences with Software Architecture Analysis of Usability. International Journal of Information Technology and Web Engineering, 3:(4),pp.1-29, (2008)
14. Folmer E., van Gurp J., Bosch J.: A Framework for Capturing the Relationship between Usability and Software Architecture. In Software Process: Improvement and Practice, Vol. 8, pp. 67-87, (2003).
15. Gamma E., Helm R., Johnson R.,Vlissides J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addisson-Wesley, ISBN 0-201-63361-2, (1995)
16. Garvin D.: What does 'Product Quality' Really Mean?. Sloan Management Review, Fall, pp. 25-45, (1984)
17. Gruber T.R.: A Translation Approach to Portable Ontologies. Knowledge Acquisition, 5:(2), pp. 199-220, (1993)
18. Guizzardi G., Baião F., Lopes M., Falbo R.: The Role of Foundational Ontologies for Domain Ontology Engineering: An Industrial Case Study in the Domain of Oil and Gas Exploration and Production. Int. J. Inf. Syst. Model. Des. 1, 2 (April 2010), pp. 1-22, (2010)
19. Hassenzahl M.: User Experience: towards an experiential perspective on product quality. In: 20th Int'l Conference of the Assoc. Francophone d'IHM; Vol. 339, pp. 11-15, (2008)
20. Heo J., Ham D-H., Park S., Song C., Chul W.: A Framework for Evaluating the Usability of Mobile Phones based on Multi-level, Hierarchical Model of Usability Factors. Interacting with Computers, Elsevier. 21:(4), pp. 263-275, (2009)
21. ISO/IEC 25010: Systems and Software Engineering - Systems and Software Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models, (2011)

22. ISO/IEC 9126-1: Software Engineering Product Quality - Part 1: Quality Model, (2001)
23. Kitchenham B., Hughes R., Linkman S.: Modeling Software Measurement Data. IEEE Transactions on Software Engineering. (27):9, pp. 788-804, (2001)
24. Law E., Abrahão S.: Interplay between User Experience (UX) Evaluation and System Development. In: Int'l. Journal of Human Computer Studies, 72(6): pp- 523-525, (2014)
25. Lew P., Qanber A. M., Rafique I., Wang X., Olsina L.: Using Web Quality Models and Questionnaires for Web Applications Evaluation. IEEE proc. QUATIC, pp. 20-29, (2012)
26. 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)
27. Lindvall M., Donzelli P., Asgari S., Basili V.: Towards Reusable Measurement Patterns, 11th IEEE Int'l Symposium in Software Metrics, pp. 1-8, (2005)
28. McGarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J., et al.: Practical Software Measurement: Objective Information for Decision Makers. Addison-Wesley Professional. ISBN-13: 978-0-201-71516-3, (2001)
29. 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)
30. Morasca S.: Software Measurement, Università dell'Insubria, Como, Italy, Reading Mater-1999, available at http://www.uio.no/studier/emner/matnat/ifi/INF5181/h11/undervisningsmateriale/reading-materials/Lecture-06/morasca-handbook.pdf, Accessed 2016-01-19
31. Nielsen, J., Budiu, R.: Mobile Usability, New Riders, Berkeley CA, (2012)
32. OMG-UML. Unified Modeling Language Specification, Version 2.0. (2005)
33. Olsina L., Santos L., Lew P.: Evaluating Mobileapp Usability: A Holistic Quality Approach. In: 14th Int'l Conference on Web Engineering, ICWE 2014, S. Casteleyn, G. Rossi, and M. Winckler (Eds.): Springer, LNCS 8541, pp. 111-129, (2014)
34. Olsina L., Lew P., Dieser A., Rivera 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)
35. 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)
36. Olsina L., Rossi G., Garrido A., Distante D., Canfora G.: Web Applications Refactoring and Evaluation: A Quality-Oriented Improvement Approach. In: Journal of Web Engineering, Rinton Press, US, 7:(4), pp. 258-280, (2008)
37. Papa M.F.: Toward the Improvement of a Measurement and Evaluation Strategy from a Comparative Study. In LNCS 7703, Springer: Current Trends in Web Engineering, ICWE Int'l Workshops, Grossniklauss M. and Wimmer M. (Eds.), pp. 189-203, (2012)
38. Rodríguez M.; Genero M.; Torre D.; Blasco B.; Piattini M. : A Methodology for Continuous Quality Assessment of Software Artefacts. 10th International Conference on Quality Software (QSIC 2010), pp. 254-261, (2010)
39. 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)