Strategy Patterns and Quality Views for Evaluating and Improving Usability

Belen Rivera[1], Pablo Becker[1], Philip Lew[2] and Luis Olsina[1]

[1]GIDIS_Web, Web Engineering School at Universidad Nacional de La Pampa, Argentina
[2]School of Computer Science and Engineering, Beihang University, China
[riveramb, beckerp, olsinal]@ing.unlpam.edu.ar, philiplew@gmail.com

Abstract. For those software organizations that frequently perform quality assurance activities devoted to measurement, evaluation (ME) and change/improvement (MEC) projects, a well-founded quality evaluation and improvement approach may be useful. In this direction, we have developed a holistic quality evaluation and improvement approach whose architecture is based on two pillars, namely: a quality multi-view modeling framework, and ME/MEC integrated strategies. In order to bolster the former pillar, in this work we specify 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 to be used in ME/MEC projects. A strategy pattern is a reusable solution to recurrent problems in MEC projects. For a specific project goal, the selected strategy pattern allows one to instantiate a concrete strategy, which embraces a set of tailored activities and methods for measurement, evaluation, analysis and change. In this paper, we also discuss a set of strategy patterns and document a concrete strategy pattern applied in one case study, which is aimed at understanding and improving Usability of a social network.

1 Introduction

Developing and managing successful software and web applications with economic and quality issues in mind is not an easy job. Software organizations that perform quality assurance activities should have a well-established quality evaluation and improvement approach to fulfill measurement, evaluation (ME) and change (MEC) project goals in a systematic and disciplined way. For instance, the quality evaluation approach should clearly establish ME/MEC strategies –which include a set of principles, activities, methods and tools- to specify quality characteristics and attributes, and collect, store, and use metrics and indicators and their values in a trustworthy manner. Furthermore, in order to make the analysis and decision-making process more robust, it is necessary to ensure that measures and indicators values are repeatable and comparable among the organization's projects [18].

For a given ME/MEC project, firstly, a concrete project goal should be established, for example: 'Understand the Usability of the XYZ system'. Then, 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 the *strategy* concept as: "*principles, patterns, and particular domain concepts and framework that may be specified by a set of tailored processes, in addition to a*

set of appropriate methods and tools as core resources for helping to achieve a project goal" [5]. In order to choose the suitable strategy from a set of ME/MEC strategies, the target quality view must be taken into account. A quality view relates accordingly an entity super-category such as product, system, system in use, with a quality focus such as internal quality (IQ), external quality (EQ), and quality in use (QinU). For the above example, the underlying quality view is the System Quality View, where System is the entity super-category to be evaluated regarding the EQ focus and the Usability characteristic. In turn, Usability is represented in an EQ model, which may combine sub-characteristics and attributes. Lastly, 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 recent years, we have developed a *holistic quality evaluation and improvement approach* [26] 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 capabilities support the principle of being integrated [29] since the same terms are consistently used for activities and methods.

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*) [27], 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 [5]. For example, a 'measurement' -from the ME domain ontology- has the semantic of 'task' -from the process generic ontology. Likewise, the 'metric' term has the semantic of 'method'; the 'measure' has the semantic of 'outcome', and so forth. In light of having a more complete conceptual base for the *holistic quality evaluation and improvement approach,* we sought the opportunity of developing an ontology for the *quality multi-view modeling framework*, i.e., for the above-mentioned (1) pillar. It is worth mentioning that quality views and their 'influences' and 'depends on' relationships are also considered by the ISO 25010 standard [16]. However, the definition of the quality view term and other related terms, are missing in this standard. Also, other works mix up the quality view concept with the quality focus concept, as we see later on. Consequently, developing an ontology of quality views can be helpful to provide an explicit semantic for this domain, in addition to strengthening our *holistic quality evaluation and improvement approach*.

On the other hand, quality views and their 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) [32], *Practical Software Measurement* (PSM) [22], and *Quality Improvement Paradigm* (QIP) [4]. 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 are also often neglected.

In the last decade, we have earned experience in developing a couple of specific ME/MEC strategies. For instance, we have developed the GOCAME (*Goal-Oriented Context-Aware Measurement and Evaluation*) and SIQinU (*Strategy for Improving Quality in Use*) strategies, which were applied in several concrete evaluation and improvement projects [20, 26, 27, 28, 29]. 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.

Recently, 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 [10, 12]. 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. To the best of our knowledge, this specific contribution fills a gap in the current literature.

Therefore, the major contributions of this research are: (i) Specify an ontology of quality views; (ii) Discuss the quality views ontology applicability for defining and selecting strategy patterns in ME/MEC projects; (iii) Analyze strategy patterns for different quality views and project goals; and (iv) Specify a concrete strategy pattern and perform its instantiation for the evaluation and improvement of the Facebook's mobileapp Usability.

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 holistic evaluation and improvement approach. Also, quality views and EQ and QinU focuses are illustrated considering the Usability and User Experience (UX) characteristics. Section 4 stresses the practical impact of the quality multi-view framework when defining and selecting strategy patterns for specific ME/MEC project goals. Also, this section documents thoroughly one strategy pattern. 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. Firstly, regarding the state-of-the-art literature we analyze the research work related to ontologies of quality views. Secondly, 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 exist works that deal with quality views and quality models. But as far as we know there is no research defining and specifying an

ontology of quality views, nor an explicit glossary of terms. One of the most relevant documents previously cited is the ISO 25010 standard, in which different quality views and their 'influences' and 'depends on' (or 'is determined by') relationships are represented informally in its Annex C. It illustrates that the software lifecycle processes (such as the quality requirements process, design process and testing process) influence the quality of the software product and the system; the quality of resources, such as human resources, software tools and techniques used for the process, influence the process quality, and consequently, influence the product quality; among other influence relationships between quality views. However, the explicit meaning of the quality view concept in [16] 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, where some views can be evaluated by means of the quality model that the standard provides.

Another initiative related to quality views is analyzed in [23] in which just the 'influences' relationship between EQ and QinU characteristics is determined by means of Bayesians networks, taking as reference the ISO 9126-1 [17] standard. However, it does not discuss a holistic quality evaluation approach that links quality views with ME/MEC strategies, as we propose. Finally, in [26] 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 [16] adding new sub-characteristics for EQ and QinU, and considers the 'influences' and 'depends on' relationships for three quality views, namely: *Software Product, System* and *System-in-Use Quality Views*. But the explicit quality view component and the included terms as we propose in this paper are missing. Also, the 2Q2U quality models were instantiated using an integrated strategy called SIQinU [20]. This strategy allows improving QinU incrementally, from the EQ improvement viewpoint. SIQinU is an instance of one of the strategy patterns that we discuss in Section 5.

Regarding the second concern, i.e., ME and MEC integrated strategies, there exists a couple of related works such as [2, 4, 22, 32]. 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 [29]. But none consider the quality views' concepts and the 'influences' and 'depends on' relationships, nor the ME/MEC strategy pattern idea. In [21], 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 [12], 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 [10, 11] 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 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 relates quality views' terms with non-functional requirements' terms as we document in Section 3.1.1. Our approach ties together quality views (entity super-categories and quality focuses) and their relationships, in addition to tailor-able 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 Approach

As indicated above, the architecture of the *holistic quality evaluation and improvement 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 Software 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, e.g., Software Product, System, System in use. Each entity super-category is evaluated considering its corresponding quality focus such as IQ, EQ, and QinU. The relationship between an entity super-category and its quality focus is called Quality View. For example, the 'System' entity super-category and the 'EQ' focus conform the 'System Quality View'. Additionally, 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 (the root characteristic such as EQ) in addition to characteristics and sub-characteristics (such as Usability and Operability) to be evaluated which combine measurable attributes. So the *quality multi-view modeling framework* embraces concepts such as quality views, quality models, 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 [5, 27]. 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 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 term among other related terms, as well as the linking with non-functional

requirement terms are missing. In order to improve these weaknesses, we have recently defined in [31] an ontology of quality views. In the present manuscript, we use the same ontology with some revised definitions of terms and relationships.

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*" [13]. On the other hand, van Heijst *et al*. [33] distinguish different types of ontologies regarding the subject of the conceptualization, e.g., domain ontologies, which express conceptualizations that are intended for particular domains; and generic ontologies, which include concepts that are considered to be generic across many domains.

Regarding the above classification, 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. Fig. 1 depicts this ontology using the UML class diagram [24] for representation and communication purposes. Also, its terms and relationships are defined in tables 1 and 2 respectively. For the construction of the ontology, we have followed the stages proposed in the METHONTOLOGY [9] 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.

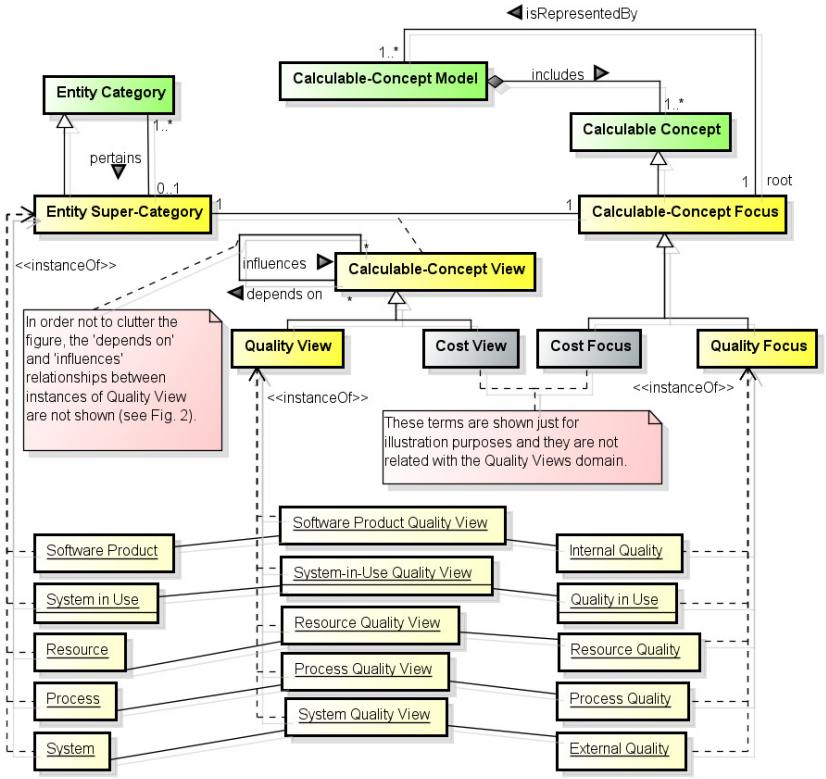

Fig. 1. Terms and some instances for the ontology of Quality Views.

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. 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 (synonym: Object Category) (from ME ontology)	Object category that is to be characterized by measuring its attributes.
Entity Super-Category	Highest abstraction level of an entity category of value to be characterized and assessed in Software Engineering organizations. Note 1: Names of entity super-categories are *Resource*, *Process*, *Software Product*, *System*, *System in use*, among others.
External Quality	It is the *quality focus* associated to the *system* entity super-category to be evaluated.
Internal Quality	It is the *quality focus* associated to the *software product* entity super-category to be evaluated.
Process	It is the *entity super-category* which embraces work definitions.
Process Quality	It is the *quality focus* associated to the *process* entity super-category to be evaluated.
Process Quality View	It is the *quality view* that relates the *process quality* focus with the *process* entity super-category.
Quality Focus	It is a *calculable-concept focus* for quality.
Quality in Use	It is the *quality focus* associated to the *system-in-use* entity super-category to be evaluated.
Quality View	It is a *calculable-concept view* for quality.
Resource	It is the *entity super-category* which embraces assets that can be assigned to processes, activities and tasks. Note 1: Examples of assets are Tool, Strategy, Software team, etc.
Resource Quality	It is the *quality focus* associated to the *resource* entity super-category to be evaluated.
Resource Quality View	It is the *quality view* that relates the *resource quality* focus with the *resource* entity super-category.
Software Product	It is the *entity super-category* which embraces software programs (i.e., source codes), specifications (i.e., requirements specifications, architectural specifications, data specifications, testing specifications, etc.), and other associated documentation.

Software Product Quality View	It is the *quality view* that relates the *internal quality* focus with the *software product* entity super-category.
System	It is the *entity super-category* which embraces software programs (i.e., applications) running in a computer environment, but not necessarily in the final environment of execution and usage.
System in Use	It is the *entity super-category* which embraces operative software applications used by real users in real contexts of use.
System-in-Use Quality View	It is the *quality view* that relates the *quality in use* focus with the *system-in-use* entity super-category.
System Quality View	It is the *quality view* that relates the *external quality* focus with the *system* entity super-category.

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*.

One core term in this ontology is *Calculable-Concept View*. This term relates the *Entity Super-Category* term with the *Calculable-Concept Focus* term. An Entity Super-Category is the highest abstraction level of an *Entity Category* to be characterized for measurement and evaluation purposes. On the other hand, a Calculable-Concept Focus is a *Calculable Concept* that represents the *root* of a *Calculable-Concept Model*.

Fig. 1 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 Fig. 1 and are not defined in Table 1. Some instances of Quality Focus are for example *Internal Quality, External Quality* and *Quality in Use*. In Table 1 we define Internal Quality as "*the quality focus associated to the software product entity super-category to be evaluated*". Also, External Quality is defined as "*the quality focus associated to the system entity super-category to be evaluated*", and Quality in Use as "*the quality focus associated to the system-in-use entity super-category to be evaluated*".

The relationship between an Entity Super-Category and its associated Quality Focus is the *Quality View* key concept of our ontology. A Quality View is a Calculable-Concept View for quality. Instances of the Quality View term are *Software Product Quality View*, *System Quality View*, *System-in-Use Quality View*, *Resource Quality View* and *Process Quality View* terms, as shown in Fig. 1. It is worth mentioning that in the figure not all instances of quality views are shown, e.g., the Service Quality View, amongst others.

Fig. 2 shows the *influences* and *depends on* relationships between instances of quality views which are commonly present in development, evaluation and maintenance projects.

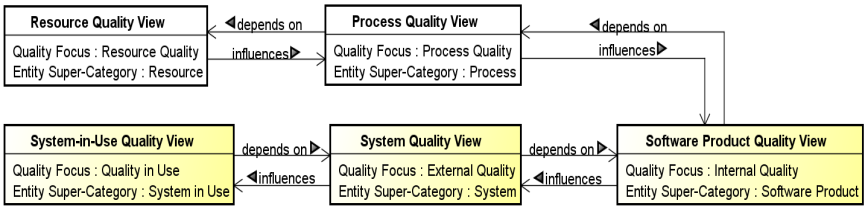

Fig. 2. An instantiation of typical quality views in software development projects.

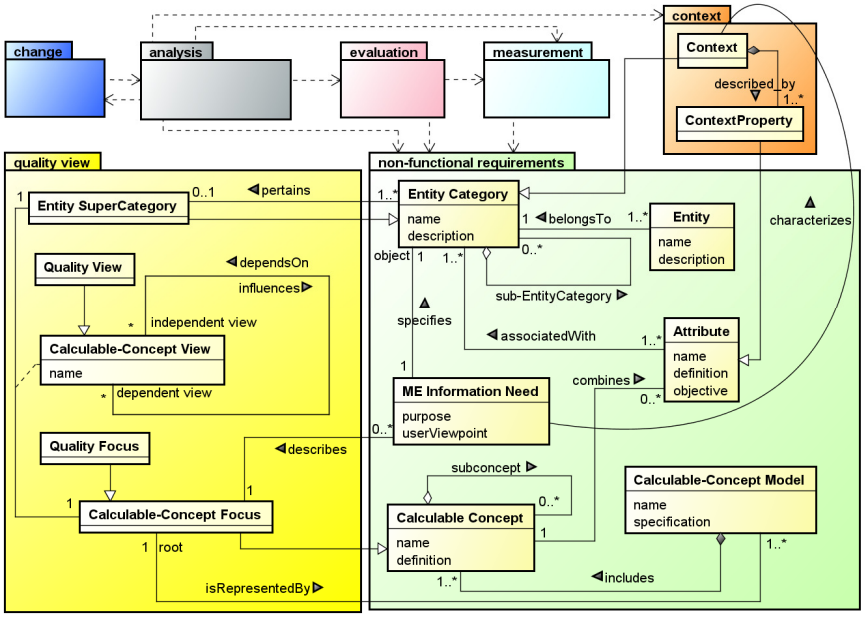

Fig. 3. The `quality_view` component which extends the C-INCAMI conceptual framework. (Note that many C-INCAMI components are drawn without terms for better visualization. In Fig. 7, the measurement and evaluation components are expanded).

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 performing. 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. On the other hand, the depends on relationship has the opposite semantic.

As commented above, the relationship between two quality views are labeled as *influences* and *depends on.* Additionally, Fig. 3 shows the corresponding *dependent* and *independent view* roles for this relation as well. Looking at Fig. 2, for example, the Resource Quality View is the *independent view* while the Process Quality View is the *dependent view*. On the other hand, the *influences* and *depends on* relationships

are transitive. Hence, it can be stated that the Resource Quality View influences the Software Product Quality View, as also suggested in the ISO 25010 standard.

Lastly, note that the quality views ontology shares some terms with the ME ontology presented in [27]. Particularly, in Fig. 3, an *Entity Super-Category* is an *Entity Category*, which is a term from the `non-functional requirements` component. Entity Category is defined in Table 1 as "*the object category that is to be characterized by measuring its attributes*". Also, a Calculable-Concept Focus is a Calculable Concept and represents the root of a Calculable-Concept Model. In Table 1 a Calculable-Concept Model is defined as "*the set of calculable concepts and the relationships between them, which provide the basis for specifying non-functional requirements and their further evaluation*".

As result, in Fig. 3, the new terms are grouped into the `quality_view` component which are linked with the former C-INCAMI `non-functional requirements` component.

3.1.2 Exemplifying Quality Views with EQ and QinU Focuses

When evaluating Usability and UX for mobile and web entities, suitable quality views should be considered [15, 25] regarding the project goal. For this aim, potential quality views are those yellow-colored in Fig. 2, namely: Software Product, System and System-in-Use Quality Views. For each quality view's focus, a quality model should be selected. A quality model specifies non-functional requirements in the form of characteristics, sub-characteristics and attributes.

For the sake of illustration, a question that a reader may ask is: to which quality focus can Usability and UX characteristics be related? The right answer to this question is that Usability can be related to the IQ and EQ focuses, while UX to the QinU focus.

In Figures 4 and 5, 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 [25, 26], which are basically an extension of the ISO 25010 models [16]. Usability is defined in Fig. 4 as "*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*". This definition 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 IQ and EQ focuses respectively.

In addition, we have defined Actual UX in Fig. 5 as "*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*" [25]. This definition considers the System-in-Use entity super-category so UX and its sub-characteristics (e.g., Satisfaction, and Actual Usability) are related to the QinU focus. As a consequence, Usability and UX are linked mainly to the three yellow-colored quality views represented in Fig. 2. That is, Usability is a characteristic related to both Software Product Quality View and System Quality View. Instead, UX is a characteristic related just to the System-in-Use Quality View. This clear separation of concerns between Usability concepts and quality views and their relationships foster a more robust evaluation and improvement approach, as we discuss later on.

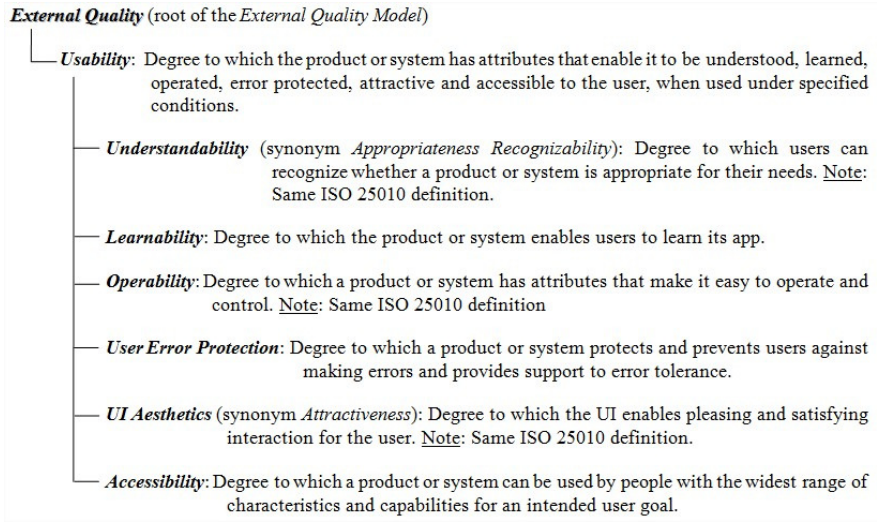

Fig. 4. Definitions of sub-characteristics related to the Usability characteristic for the EQ focus.

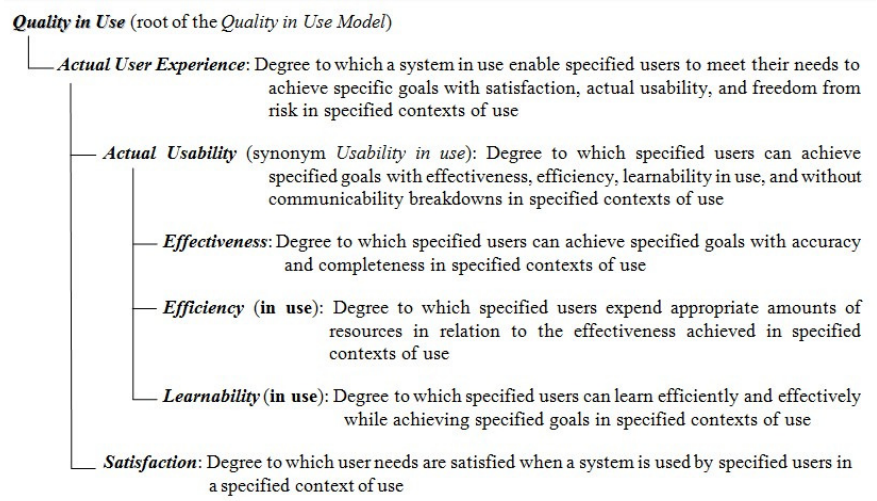

Fig. 5. Definitions of some characteristics and sub-characteristics for the QinU focus.

Fig. 6 illustrates two target entity super-categories (System and System in Use) and their corresponding quality focuses (EQ and QinU) with some quality characteristics. This rough schema is derived from some components of Fig. 3. For the 'System' entity super-category (System box in Fig. 6) 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.

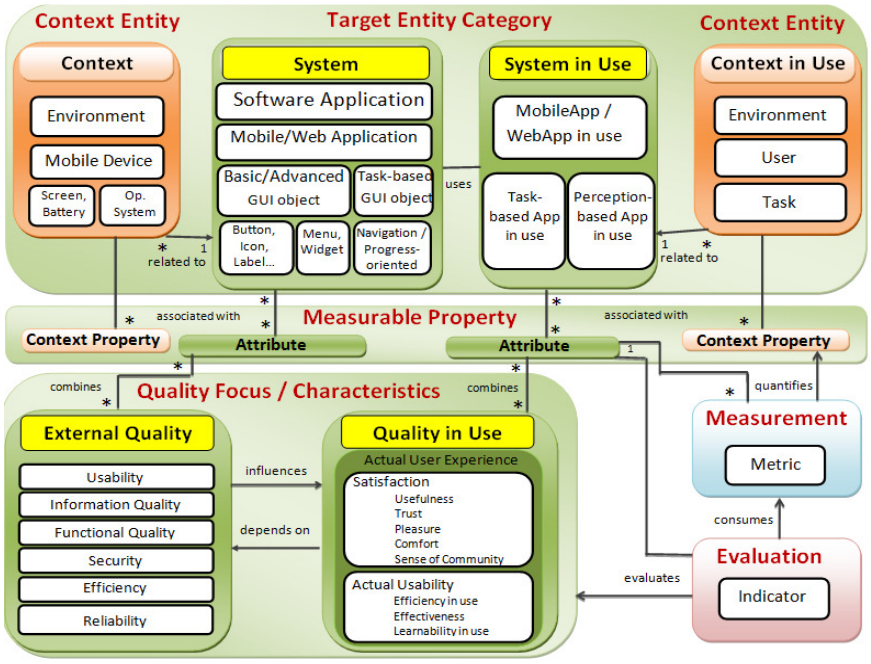

Fig. 6. Rough schema derived from Fig. 3, which relates the main building blocks for Target Entity (mainly related to GUI objects for System) and Context Entity Categories, Quality Focuses (just for EQ and QinU), Measurable Properties, Measurement and Evaluation.

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. Fig. 6 illustrates the link between Measurable Properties –attributes-, Measurement (light-blue box) and Evaluation (pink box). Therefore, 'Usability', a characteristic for the 'EQ focus', combines a set of attributes for evaluating GUI objects. For the EQ focus other characteristics (and attributes associated to other System sub-entities) such as 'Information Quality', 'Security', among others, can be used for ME purposes.

Fig. 6 also depicts the 'System in Use' box (i.e., the target entity super-category) with the corresponding 'QinU' box (i.e., the quality focus and the UX characteristic and sub-characteristics). Looking at the first 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 Fig. 5), 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', and so on.

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, Fig. 6 shows that the QinU focus (System-in-Use Quality View) depends on the EQ focus (System Quality View). Considering some empirical observations made in [20], we can indicate for instance that Actual Usability depends not only on EQ Usability, but also on other sub-characteristics such as Information Quality and Efficiency.

Lastly, Fig. 6 depicts two Context boxes, and Fig. 3 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. We discussed in [25] 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. For instance, System in Use is characterized by a 'Context-in-Use' entity, which in turn can aggregate 'Environment', 'User' and 'Task' sub-entities, while 'Context' for System can be characterized by sub-entities such as 'Device', 'Screen', 'Operating System', etc. To describe the Context, *Context Properties* (Fig. 3) 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 our approach, an integrated strategy simultaneously supports three capabilities [29]: a *domain conceptual base and framework*, *process perspective specifications*, and *method specifications*. Regarding the first capability, Fig. 3 shows the C-INCAMI conceptual base and framework, which 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, interdependencies, artifacts, roles, and so forth. Additionally, process specifications can consider different process perspectives such as functional, behavioral, informational and organizational [8]. Usually, process specifications primarily 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/EQ/QinU evaluation and improvement cycles, starting evaluations from Task-based and/or Perception-based App-in-use entities –both illustrated in Fig. 6-, and their corresponding characteristics and attributes. So SIQinU embraces two quality views (System-in-Use Quality View

and System Quality View) and explores the aforementioned relationships between views.

Due to the experience gained in the last decade in the development and use of the above strategies [19, 20, 25, 28, 29], 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 one strategy pattern.

To summarize, our holistic quality evaluation and improvement approach can help software 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 a project goal.

4 Documenting the GoMEC_1QV Strategy Pattern

In Software 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*" [12]. 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 types of evaluation purposes so a suitable strategy to achieve a given goal should be selected. That is, the strategy should be the most suitable one for the intended goal. Strategy patterns arise as a way for providing a solution in the instantiation of ME/MEC strategies considering the project goal and the number of included quality views.

We have specified a set of strategy patterns [30], following to some extent the pattern specification template used in [12]. 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 the pattern 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 worth to remark 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 of strategy patterns and their instantiation regarding different ME/MEC project goals.

If a project has a goal such as 'improve the usability of the XYZ mobile application', this means that it has an 'improve' purpose which embraces the *System Quality View*. This is because the concrete *Entity* is the 'XYZ mobile application' which belongs to the System *Entity Super-Category*, and the *Quality Focus* is EQ (recall Fig. 4, where Usability is the characteristic linked to the EQ focus). For this goal, it is necessary not only to understand the current situation of the entity but also to perform changes on it in order to re-evaluate it and gauge the improvement gain.

Another project goal can have the same 'improve' purpose, even though the quality view involved could be a different one (e.g., Software Product Quality View, Process Quality View, or System-in-Use Quality View). Hence, for a concrete MEC project embracing one quality view, the same MEC strategy pattern should be selected and tailored accordingly. That is, for the intended quality view, the pattern (its processes and methods) should be instantiated appropriately. This strategy pattern is the so-called *Goal-oriented Measurement, Evaluation and Change for One Quality View* (alias GoMEC_1QV).

The items used to specify the pattern are the following:

Intent: To provide a solution in the instantiation of a measurement, evaluation, analysis and change strategy aimed at supporting a specific improvement goal of a project 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 improve 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 main structural aspects of the three strategy capabilities.

I. The concepts in the `non-functional requirements`, `context`, `measurement`, `evaluation`, `change`, and `analysis` components (Fig. 3 and 7) are defined as sub-ontologies. The included terms, attributes and relationships belong to the MEC area. In Fig. 7 we show just the main ME terms. Note that ME terms in Fig. 7 are also enriched with terms from a generic process ontology [5] by means of stereotypes. These concepts are used consistently in the activities, artifacts, outcomes and methods of any ME/MEC strategy.

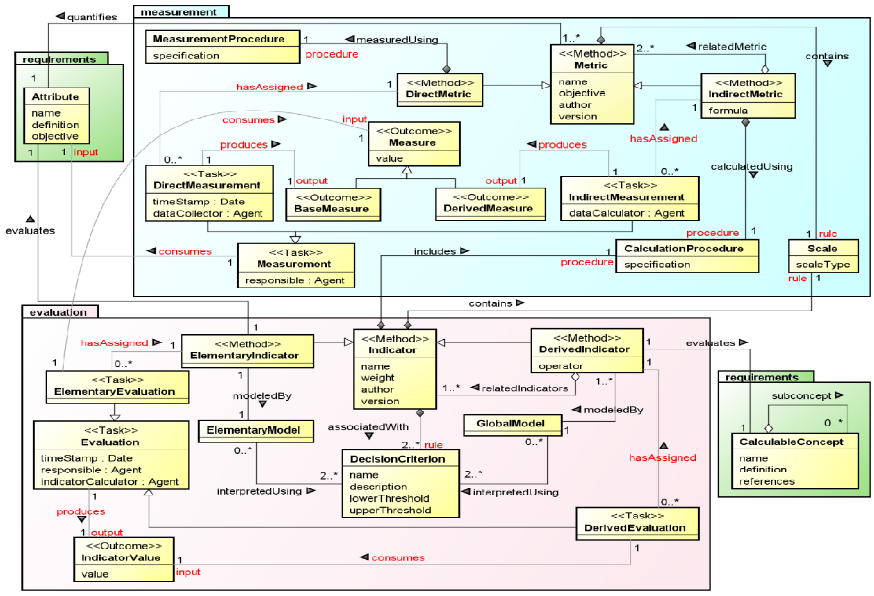

Fig. 7. `Measurement` and `evaluation` components of the C-INCAMI conceptual framework enriched with process terms.

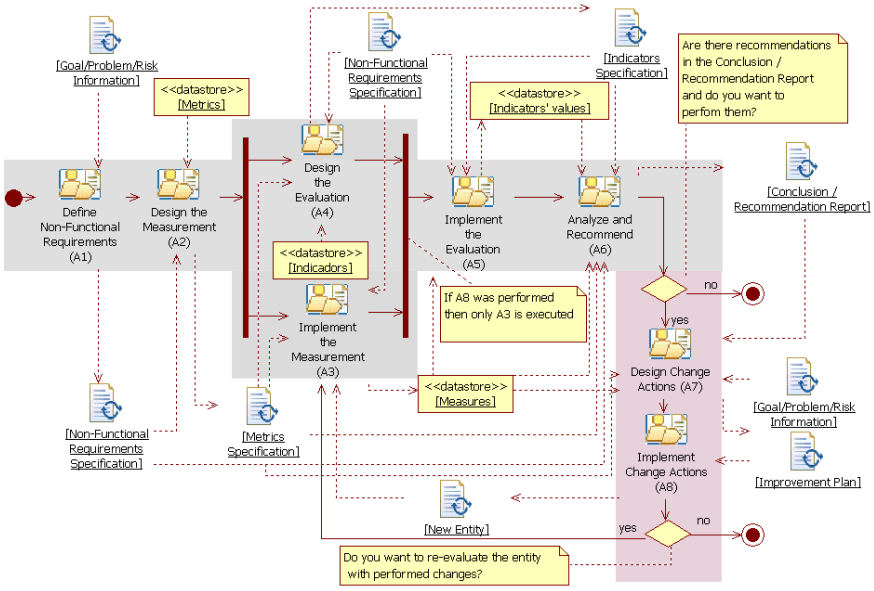

Fig. 8. Generic process from the functional and behavioral perspectives for the GoMEC_1QV pattern.

II. The process specification is made 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, Fig. 8 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. Table 3 exemplifies two method specification templates: one for a direct metric used as method specification for direct measurement tasks; and the 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.

Table 3. Method specification template for: a) Direct Metric and b) Elementary Indicator.

a) Direct Metric
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: Info about the used tool if any)

b) Elementary Indicator
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:

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 [28]. Besides, this pattern was instantiated in a MEC project for the Resource Quality View [29].

Scenario of use: The present example stems from the Facebook's mobileapp (v3.8 for Android) Usability case study performed in Dec. 2013 [25], and then replicated for the v14 in Sept. 2014, applying also the same Usability requirements.

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 'EQ' where the evaluated characteristic (Calculable Concept) is 'Usability' and its related 'Understandability', 'Learnability', 'Operability', 'User Error Protection' and 'UI Aesthetic' sub-characteristics.

For the above project goal, the GoMEC_1QV strategy pattern should be selected and instantiated for the 'System Quality View'. Therefore, the 8 generic activities that the pattern provides as a solution (Fig. 8) are instantiated. E.g., 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 8 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' (Fig. 9), and the 'Requirements Tree for EQ' artifacts. In Table 4, 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
Entity Category: Social Network Application **Entity Super-Category:** System
(Concrete) Entity: Facebook mobile app, v14 for Android
Quality Focus*:* External Quality
Context Properties: *Mobile device type*: "Mobile phone", *Screen size*: "540x960px/4.3inches", *Mobilephone generation*: "Full-sized smartphone", amongst others.
Calculable Concepts (Characteristics): Usability and its related sub-characteristics: Understandability, Learnability, Operability, User Error Protection, and User Interface Aesthetics

Fig. 9. Summarized *Information Need* artifact produced in the instantiated A1 activity.

Table 4. Definition of EQ/Usability sub-characteristics and attributes –in *italic*.

Characteristic/*Attribute*	**2Q2U v2.0 Definition**
1 Usability	See the definition in Figure 4.
1.1 Understandability (synonym Appropriateness Recognizability)	See the definition in Figure 4.
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	See the definition in Figure 4.
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	Degree to which meaningful error messages are provided upon invalid

appropriateness	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 4.
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 provide 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 provide 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 provide 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) are 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 are 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 4.
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 (synonym Attractiveness)	See the definition in Figure 4.
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 ('Design the Measurement for EQ') activity, metrics are selected and assigned to quantify all attributes of the requirements tree. Metrics are retrieved from a repository (Metrics <<datastore>> in Fig. 8) 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 4). 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*". Fig. 10 shows full details of the metric specification.

Concrete Entity: **Name**: *Facebook app*; **Version**: 14 (Set 2014); **Sub-Entity Description:** Set of *Screens* of the *Facebook app* where the *Main controls bar* is (or should be) containing the set of *Main controls (Buttons)*
Attribute: **Name**: *Permanence of main controls*; **Code**: *1.3.3.1.1* in Table 4

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: $\%MCP = \left[\frac{\sum_{i=1}^{m}\sum_{j=1}^{n} MCPLij}{(m*n)}\right] * 100$; for i=1 to m and j=1 to n, 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**: 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

Fig. 10. Indirect and direct metric specifications for the *Permanence of main controls* attribute.

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:** %

Fig. 11. Elementary Indicator specification for the Permanence of main controls attribute.

(A3) Implement the Measurement for EQ: The pattern's generic process establishes A3 as the next activity to be instantiated. 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, A3 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 <<datastore>>.

Table 5. Requirements Tree for EQ with evaluation outcomes yielded in the 2013 and 2014 studies. (Note: EI means Elementary Indicator; DI means Derived Indicator).

Characteristic /Sub-characteristic / *Attribute*	**2013**		**2014**	
	EI	**DI**	**EI**	**DI**
1 Usability		**60.5**		**65.1**
1.1 Understandability		**76.1**		**89.9**
1.1.1 Familiarity		76.1		89.9
1.1.1.1 Global organization scheme understandability	100		100	
1.1.1.2 Control icon ease to be recognized		65.2		74.8
1.1.1.2.1 Main control icon ease to be recognized	60		71.4	
1.1.1.2.2 Contextual control icon ease to be recognized	87.5		88.9	
1.1.1.3 Foreign language support	0		100	
1.2 Learnability		**59.7**		**66.4**
1.2.1 Feedback Suitability		75.9		85.6
1.2.1.1 Current location feedback appropriateness	52.6		80.3	
1.2.1.2 Alert notification feedback appropriateness	100		100	
1.2.1.3 Error message appropriateness	75		75	
1.2.2 Helpfulness		45		49.1
1.2.2.1 Context-sensitive help appropriateness	50		54.5	
1.2.2.2 First-time visitor help appropriateness	0		0	
1.3 Operability		**80.7**		**80.4**
1.3.1 Data Entry Ease		90		90
1.3.1.1 Defaults	100		100	
1.3.1.2 Mandatory entry	50		50	
1.3.1.3 Widget appropriateness	100		100	
1.3.2 Visibility (synonym **Optical Legibility**)		81.5		81.5
1.3.2.1 Color visibility appropriateness		100		100
1.3.2.1.1 Brightness difference appropriateness	100		100	
1.3.2.1.2 Color difference appropriateness	100		100	
1.3.2.2 GUI object size appropriateness		63		63
1.3.2.2.1 Control (widget) size appropriateness	100		100	
1.3.2.2.2 Text size appropriateness	42.1		42.1	
1.3.3 Consistency		75.5		74.6
1.3.3.1 Permanence of controls		57.3		55.7
1.3.3.1.1 Permanence of main controls	54.9		46.2	
1.3.3.1.2 Permanence of contextual controls	67.4		100	
1.3.3.2 Stability of controls	95.5		95.5	
1.4 User Error Protection		**8.4**		**8.4**
1.4.1 Error Management		8.4		8.4
1.4.1.1 Error prevention	0		0	
1.4.1.2 Error recovery	16.7		16.7	
1.5 UI Aesthetics		**80.8**		**91.1**
1.5.1 UI Style Uniformity		80.8		91.1
1.5.1.1 Text color style uniformity	85		95	
1.5.1.2 Aesthetic harmony	79		89.5	

(A4) 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. Fig. 11 shows the elementary indicator specification for the attribute *Permanence of main controls*, following the elementary indicator specification template shown in Table 3 b).

(A5) Implement the Evaluation for EQ: From the measures obtained in A3 and using the selected elementary indicators and derived indicators designed in A4, 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 4th and 5th columns of Table 5, for the re-evaluation performed in Sept. 2014.

Notice that in Table 5, 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.

(A6) Analyze and Recommend for EQ: After finishing ME activities, the instantiated A6 ('Analyze and Recommend for EQ') activity should be performed. Table 6 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 6, 'H' means high priority. Recommendations enable to design and perform change actions for improvement.

Looking at the elementary indicator values for 2014 in Table 5, 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 6 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. Fig. 12 shows two selected screenshots belonging to the set of appropriate screens that were evaluated (34 out of 38) for the Facebook mobile app (v14 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.

(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 7 shows a fragment of the 'Improvement Plan for EQ' artifact. Basically, per each recommendation (R) in Table 6, 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.

Table 6. Fragment of the Recommendation Report for EQ artifact generated in the instantiated A6 activity. (Note: H stands for High, 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

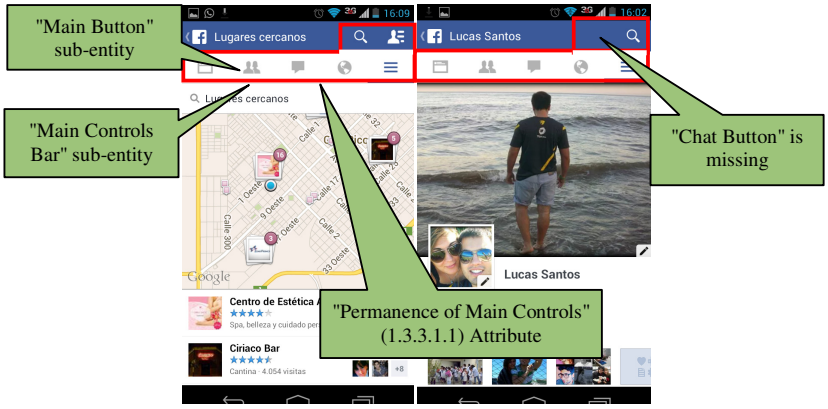

Fig. 12. Two Facebook v14 screenshots: Left one shows the 'Main Controls Bar' sub-entity, which is composed of seven 'Main Buttons'; Right one highlights the missing 'Chat button'.

Table 7. 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

For the 1.3.3.1.1 attribute, the change action is CA2.1 in Table 7, derived from R2.1 in Table 6. 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 [28]). 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, A3, A5 and A6 must be performed again, as specified in Fig. 7. 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.

5 Discussion of other ME/MEC 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 the entity, re-evaluate it, and gauge the achieved improvement gain. 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 social network app. Usability was linked in this case to the EQ focus. However, we may also instantiate this pattern to improve, for instance, the Usability regarding the IQ focus. The sub-entity category for the IQ focus can be an artifact at early stages of development, such as an architectural design, as presented in [11].

Another strategy pattern is the *Goal-oriented Measurement, Evaluation, and Change for Two Quality Views* (alias GoMEC_2QV). This pattern gives a solution for an improvement project goal which involves two quality views and their relations. Recall that between two quality views, the 'influences' and 'depends on' relationships can be used. This implies that one quality view plays the role of *dependent quality view*, while the other plays the role of *independent quality view* (see these roles in Fig. 3).

For example, if we consider the System Quality View and the System-in-Use Quality View, these relations embrace the hypothesis [16] that evaluating and improving the EQ 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 EQ 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 Software Product Quality View. That is to say, by

evaluating and improving the resource quality can be one means for improving the IQ focus of a product. For example, 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 any 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 Fig. 8. This is the simplest pattern to be instantiated. Also, so far, it is the mostly used in the ME projects we have performed, e.g., in the evaluation of a mash-up application [26] and a shopping cart [27]. This pattern can be used to evaluate not only any quality focus depicted in Fig. 2 but also 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 of Fig. 2 can be evaluated. Both Usability (for IQ and EQ) and UX (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 a XYZ mobile application by means of improving the system and software product quality". The specification of GoMEC_3QV is also 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 [7] 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 Fig. 1.

6 Conclusions and Future Work

In this work, we have enhanced our *holistic quality evaluation and improvement approach* by strengthening its architecture based on two pillars. Firstly, 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 semantic processability. In the process of building this ontology, we have reviewed related literature about quality views and multi-view modeling frameworks. Specifically, we have discussed in Section 2 that an ontology, taxonomy or glossary of terms for this domain does not exist based on the research we have done.

Secondly, 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 described the GoMEC_1QV strategy pattern and the instantiation of it for the Facebook's mobileapp Usability case study.

As future lines of research, we will instantiate the GoMEC_3QV strategy pattern, for Usability and Security issues regarding architectural design artifacts and the IQ focus as well. This will allow us to gauge improvements in QinU for both Satisfaction and Actual Usability characteristics, taking also to some extent the Folmer *et al*. work [11] into account. We think that by using a couple of suitable strategy patterns and our evaluation and improvement approach, many of the raised issues in [10] can be covered appropriately. On the other hand, we will work on the specification and instantiation of the strategy pattern related to the evaluation, comparison and selection of competitive entities, which was mentioned in Section 5.

With respect to the quality views ontology, we are aware that in its current version both generic and quality domain-specific terms are embedded. We are planning to re-design the ontology for decoupling foundational (generic) ontology concerns from the current version, due to the relevance of this in the development of domain ontologies, as proposed in [14]. The re-design will enable reuse of the foundational ontology in specific domains such as quality and cost, amongst others.

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

Acknowledgments. 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 R., 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. 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)
6. 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)
7. CMMI (Capability Maturity Model Integration), Version.1.3. CMU/SEI-2010-TR-033, USA, (2010)
8. Curtis B., Kellner M., Over J.: Process Modelling. Communications of ACM, 35:(9), pp.75-90, (1992)
9. Fernández-López M., Gómez-Pérez A., Juristo N.: METHONTOLOGY: From Ontological Art Towards Ontological Engineering. Spring Symposium on Ontological Engineering of AAAI, pp. 33–40, Stanford University, California, (1997)
10. 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)
11. Folmer E., van Gurp J., Bosch J.: A framework for capturing the relationship between usability and software architecture. In Software Process: Improvement and Practice, 8, pp. 67–87, (2003)
12. Gamma E., Helm R., Johnson R.,Vlissides J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addisson-Wesley, ISBN 0-201-63361-2, (1995)
13. Gruber T.R.: A Translation Approach to Portable Ontologies. Knowledge Acquisition, 5:(2), pp. 199-220, (1993)
14. 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)
15. 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)
16. ISO/IEC 25010: Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, (2011)
17. ISO/IEC 9126-1: Software Engineering Product Quality - Part 1: Quality Model, (2001)
18. Kitchenham B., Hughes R., Linkman S. Modeling Software Measurement Data. IEEE Transactions on Software Engineering. (27):9, pp. 788-804, (2001)
19. 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)
20. 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)
21. Lindvall M., Donzelli P., Asgari S., Basili V.: Towards Reusable Measurement Patterns., 11th IEEE Int'l Symposium in Software Metrics, pp. 1-8, (2005)
22. 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)
23. 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)
24. OMG-UML. Unified Modeling Language Specification, Version 2.0. (2005)
25. 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)
26. 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)
27. 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)
28. 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)
29. 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)
30. Rivera M.B., Becker P., Olsina L.: Strategy Patterns for Measurement, Evaluation And Improvement Projects (In Spanish). XVIII Iberoamerican Conference in Software Engineering (CIbSE'15), Lima, Perú, pp. 166-180, (2015)
31. Rivera B., Becker, P., Olsina L.: Extending the Conceptual Base for a Holistic Quality Evaluation Approach. 1st Argentine Symposium on Ontologies and their Applications (SAOA'15) Rosario, Argentina, ceur-ws.org/Vol-1449/, pp. 121-130, (2015)
32. Rodríguez, M.; Genero, M.; Torre, D.; Blasco, B.; Piattini, M., A Methodology for Continuos Quality Assessment of Software Artefacts , 10th International Conference on Quality Software (QSIC 2010), pp.254-261, (2010)
33. 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)