Strategy Patterns 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
belenrs@yahoo.com, [beckerp, olsinal]@ing.unlpam.edu.ar, philiplew@gmail.com

Abstract. Patterns have had significant impact in many disciplines, particularly in software and web engineering, and we believe that they also provide a basis for selecting evaluation strategies via practical tips and tricks that can be easily adopted for evaluation and change projects. In this paper, we propose a holistic quality evaluation approach for usability and user experience (UX), which relies on quality views and strategy patterns. A quality view relates accordingly an entity super-category (e.g., product, system, system in use) with a quality focus (e.g., internal quality, external quality, quality in use). Usability and user experience are higher-level characteristics that should be linked to quality views appropriately. Also quality views support 'influences' and 'depends on' relationships between them. With a concrete evaluation or change project goal, our approach selects and instantiates a suitable strategy from a set of strategy patterns. Practical use of our approach is demonstrated through the specification and use of a strategy pattern in the evaluation of the Facebook mobile app.

1 Introduction

It is well known that Usability and UX are significant quality characteristics of web and mobile applications and thus their evaluations are becoming increasingly important in the mobile gadgetry industry as well. Many researchers, guidelines and standards have studied or dealt with Usability and UX such as [5, 7, 10, 11, 12, 16], to quote just a few. Some works deal with Usability attributes and guidelines [7, 16] for products at early stages of development such as user interface designs, architectural designs, task scenarios and usability patterns, etc. This quality focus for these artifacts or intermediate products is commonly named Internal Quality (IQ) [12]. Other works deal with Usability attributes and heuristics [11, 16] when the product itself is already a system –in execution state- but typically not used by real users in actual contexts of use. This quality focus for system is commonly named External Quality (EQ) [12]. Also other works [5, 10] deal with Usability in use and UX, i.e., when the system is used and perceived by real users in real contexts of use. This quality focus for system in use is commonly called Quality in Use (QinU) [12].

Looking at the state-of-the-art literature we found that Usability, Usability in use and UX are poorly linked to target entity categories (e.g., Product, System and System in use) and their contexts with quality focuses (e.g., IQ, EQ and QinU) and their quality models [17]. A *quality view* relates accordingly an entity super-category with a quality focus. Also quality views support *influences* and *depends on* relationships. Moreover, a *quality model* represents the quality focus by means of a set of characteristics, sub-characteristics and attributes like in ISO 25010 [12] and 2Q2U

(*Internal/External Quality, Quality in Use, Actual Usability, and UX*) [18] quality models. Thus, Usability and UX sub-characteristics and attributes, as non-functional requirements to be evaluated are often included in quality models.

On the other hand, quality views and their relationships are paramount for defining measurement, evaluation (ME) and change (MEC) strategies and *strategy patterns* to be used as resources in ME/MEC projects. Strategy has many definitions, but typically involves setting goals, establishing actions (tasks, activities, processes) to achieve the goals, and mobilizing resources to execute the actions. We define the term strategy 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 achieving a project goal* "[4].

In the last two lustrums, we have earned experience in developing specific strategies. For instance, we have developed 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 [13, 18, 19, 20]. Regarding the above strategy definition, both share the principle of three integrated capabilities [4], namely: the *ME domain conceptual base and framework*, the *process perspective specifications*, and the *method specifications*. However, we recently have envisioned the idea of packaging the earned experience into strategy patterns, considering the *holistic quality evaluation approach* which was based on *quality views* and *concrete strategies*.

It is recognized that patterns have had and still have significant impact in software and web engineering [7, 8]. In a nutshell, the pattern's main aim is to provide a general and reusable solution to a recurrent problem. In this sense, 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 instantiable solutions for evaluating and improving the quality focuses for different entities in MEC projects. Also, we think this work fills an important gap in the current literature.

Therefore, the major contributions of this research are: (i) *Add a quality view component* to the ME conceptual framework [19], which allows instantiating quality views and quality models accordingly; (ii) *Analyze strategy patterns for different quality views and project goals* such as evaluation and improvement for Usability and UX; and (iii) *Specify a concrete strategy pattern* and perform its instantiation for the Facebook's mobileapp Usability case study.

Following this introduction, Section 2 describes related work addressing mainly research that deals with quality views and strategy patterns. Section 3 analyzes the proposed holistic quality evaluation approach on the basis of quality views, evaluation and improvement strategies. Section 4 specifies the MEC strategy pattern for one quality view and illustrates its instantiation. Section 5 discusses other strategy patterns and, finally, Section 6 draws our main conclusions and outlines future work.

2 Related Work

In this work, we present ME/MEC strategy patterns in the context of a *holistic quality evaluation approach* whose architecture is based on two pillars, namely: (1) *a quality multi-view modeling framework;* and, (2) *ME/MEC integrated strategies*. In order to enhance the second pillar, we also developed a set of strategy patterns. Some of the existing research take these two concerns into account to an extent, but none have intertwined and integrated a single and systematic approach.

Regarding the first pillar, there exists research that deals with quality views and quality models. An example is the standard ISO 25010 [12], where different quality views and their 'influences' and 'depends on' relationships are presented. However, the explicit meaning of the quality view concept is missing. Rather, it outlines quality views in the context of a system quality lifecycle model, where each view can be evaluated by means of the quality models that the standard propose. Another initiative related to quality views is analyzed in [15] 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 standard superseded by [12]. However, it does not discuss a holistic quality evaluation approach that links quality views with ME/MEC strategies, as we propose. Finally, in [18] the 2Q2U quality framework is proposed. This framework extends the quality models defined in [12] and considers the 'influences' and 'depends on' relationships for three quality views. But the explicit *quality view* component as we propose in this paper is missing. Also, the 2Q2U quality models were instantiated using an integrated strategy called SIQinU [13]. SIQinU is an instance of one of the strategy patterns we propose in Section 5.

Regarding the second pillar, i.e., ME and MEC integrated strategies, there exists a couple of related works. For example, in [3] the GQM$^+$Strategies, which is built on top of the so-called GQM (*Goal Question Metric*) strategy is presented. Both strategies include the principle of the three integrated capabilities as indicated in the previous Section. But no one considers the quality view concepts and framework, nor the 'influences' and 'depends on' relationships, nor the strategy pattern idea either. In [14], 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. But their specifications follows no recommended form [8] such as name, intention, problem, solution/structure, known uses, etc.

Frequently, many researches that deal with patterns are mainly intended for early stages of development and change, focusing for instance on usability patterns and user interface designs, or architectural designs, etc., but seldom intended for evaluation and improvement stages in which quality views and MEC strategy patterns are considered appropriately. For example, in [7] authors define a framework that expresses relationships between *Software Architecture and Usability* (SAU). The SAU framework consists of the following concepts: (i) *Usability attributes* such as Learnability, Efficiency of use, Reliability in use and Satisfaction; (ii) *Usability properties*, i.e., a number of properties that embody heuristics and design principles that researchers in the usability field consider to have a direct positive influence on system usability; and (iii) *Architecturally sensitive usability patterns*, i.e., a number of usability patterns that authors have been identified that should be applied during the design of a system's software architecture. 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, as we discuss later on.

Moreover, some works propose pattern languages, such as in [2]. In this work, authors evaluate the usability of domain specific languages. This pattern language is composed by 17 patterns which represent a collection of usability-oriented best practices collected from a wide set of domains, i.e., from the general purpose languages design to the HCI areas. However, the two concerns we raised at the beginning of this section are not addressed. On the other hand, design patterns have also been used to help novice evaluators to find and reuse expert knowledge in order to perform usability evaluations. For example, authors in [9] propose the DEPTH method and tool, where the expert knowledge is stored in the form of design patterns. In summary, there does not exist the utilization of ME/MEC strategy patterns as we propose. Our approach ties together quality views (entities and quality focuses), relations between views, characteristics (Usability, UX, etc.) and attributes related to quality focuses, in addition to suitable strategies for measurement, evaluation, analysis and improvement, which are packaged into strategy patterns aimed at easing the strategy instantiation for different recurrent ME/MEC problems.

3 Foundations for the Holistic Quality Evaluation Approach

As commented above, the architecture of our *holistic quality evaluation approach* is built on two pillars. In sub-section 3.1, we discuss the *quality view modeling framework* which allows instantiating Product, System, System-in-Use Quality views, among others. Also, the specification of non-functional requirements, such as Usability and UX can be represented by a quality model accordingly regarding the quality view to be evaluated. In sub-section 3.2, we analyze what an *integrated strategy* for the purpose of evaluation and improvement is.

3.1 Quality View Modeling Framework

In a given ME/MEC project one or more entity super-categories can be chosen as, for example, System and System in use, which can be evaluated by means of the EQ and QinU focuses, respectively. The quality focus is the highest abstraction level concept of a quality model. Next, we represent the *quality multi-view modeling framework* as an added new for the previously developed C-INCAMI (*Contextual Information Need, Concept Model, Attribute, Metric and Indicator*) conceptual framework [19].

Fig. 1 shows the new *quality_view* component which allows the instantiation of quality views by means of the *QualityView* concept. Examples of concrete quality views are depicted in Fig. 2. A quality view is the association between an entity super-category (*EntitySuperCategory*) and a quality focus (*QualityFocus*). Moreover, the added component allows representing two relations between quality views. For instance, Fig. 2 shows that the System Quality view *influences* the System-in-Use Quality view and, in turn, the latter *depends on* or is determined by the former.

Looking at Fig. 1, a concrete *Entity* (e.g., named "Facebook mobile app") belongs to the *EntityCategory* named "Social network app", which in turns pertains to the *EntitySuperCategory* named "System". On the other hand, a *QualityFocus* is the highest level *CalculableConcept* such as "External Quality". Therefore, the *Quality View* named "System Quality" associates the "EQ" focus with the "System" entity super-category, as shown in Fig. 2.

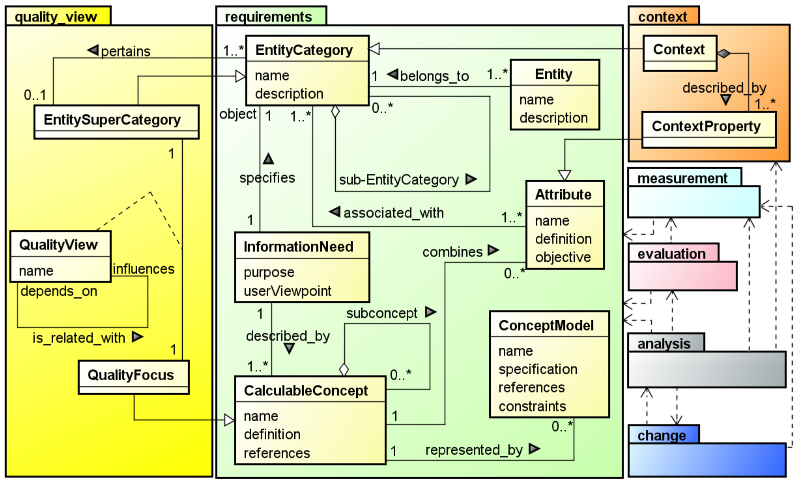

Fig. 1. C-INCAMI conceptual framework enhanced with the *quality_view* component

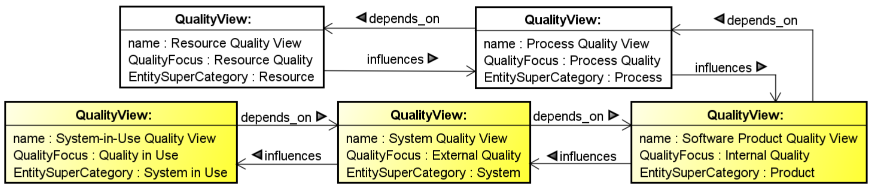
QualityView:
name : Resource Quality View
QualityFocus : Resource Quality
EntitySuperCategory : Resource
◀depends_on
influences ▶
QualityView:
name : Process Quality View
QualityFocus : Process Quality
EntitySuperCategory : Process
◀depends_on
influences ▶
QualityView:
name : System-in-Use Quality View
QualityFocus : Quality in Use
EntitySuperCategory : System in Use
depends_on ▶
◀influences
QualityView:
name : System Quality View
QualityFocus : External Quality
EntitySuperCategory : System
depends_on ▶
◀influences
QualityView:
name : Software Product Quality View
QualityFocus : Internal Quality
EntitySuperCategory : Product

Fig. 2. An instantiation of typical quality views from the *quality_view* component

Fig. 2 shows the *depends on* and *influences* relationships for typical quality views that can intervene in software production lines for development, evaluation and maintenance. It shows that the Resource Quality view –where a resource entity can be an agent, method, strategy, etc.- *influences* the Process Quality view. In turn, the Process Quality view *influences* the Software Product Quality view –where a product entity category can be a graphical user interface (GUI) design, or a source code, etc. A Product Quality view *influences* the System Quality, and in turn this *influences* the System-in-Use Quality view –being an example of this entity category, a social network mobile app in use, which is used by actual users in specific contexts of use. The *depends on* relation has the opposite semantic. Quality views are paramount for defining ME/MEC strategies and patterns, as we discuss in Sections 4 and 5.

3.1.1 Relating Usability and UX with Quality Views.

When evaluating Usability and UX characteristics for mobile and web entities, suitable quality views should be considered [11, 17]. For this aim, potential views are those yellow-colored quality views in Fig. 2, namely: Software Product, System and System-in-Use Quality views. For each quality view, and particularly for its quality focus, a quality model should be selected. A quality model specifies non-functional requirements in the form of characteristics and attributes. A question that a reader might ask is: to which quality focus can Usability and UX be related? In short, Usability can be related to the IQ and EQ focuses, while UX to the QinU focus [5].

We have defined in [17] the Usability, Actual Usability (Usability in use) and UX concepts that are included in the 2Q2U v2.0 quality models [18], which are basically an extension of the ISO 25010 models. Usability is the "*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 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. On the other side, we have defined Actual UX 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.*" [17]. This definition 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 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 the Product and System Quality views, and UX 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. 3 illustrates two target entity super-categories (System and System in Use) and their corresponding quality focuses (EQ and QinU) with some characteristics. This rough schema is derived from some components of Fig. 1. For the "System" entity super-category (System box, in Fig. 3) another sub-entity categories are identified such as "Mobile/Web Application" which in turn, from the GUI standpoint, can be subdivided in "Basic/Advanced GUI objects", and "Task-based GUI objects".

Notice that an entity cannot be measured directly. Rather, it is measured by means of associated attributes. Attributes are quantified by metrics during the measurement process and are interpreted by indicators during the evaluation process. Fig. 3 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 "Security", "Efficiency", among others, can be used for ME purposes.

Fig. 3 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 its included characteristics such as UX, among others). 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, 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.

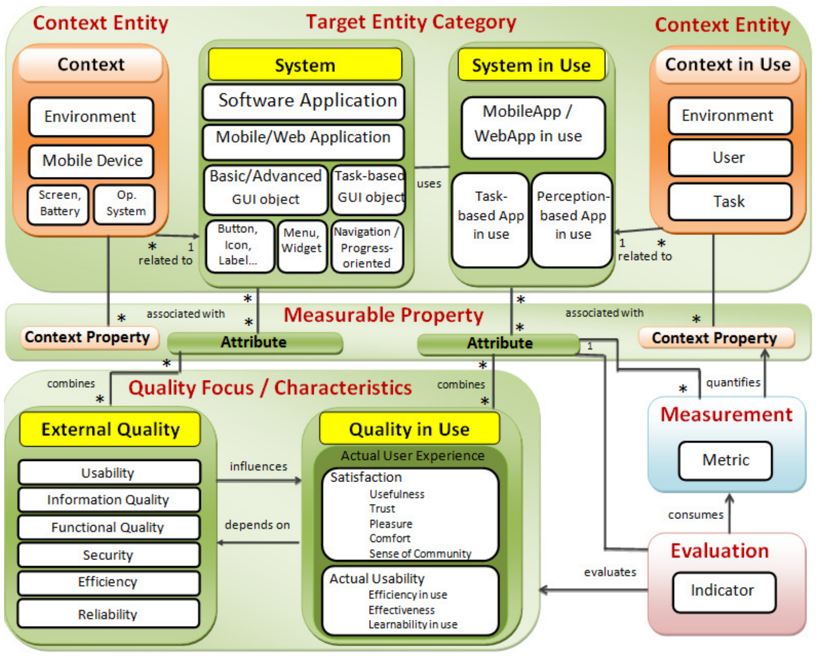

Fig. 3. Rough schema derived from Fig. 1, which relates the main building blocks for Target Entity (mainly related to GUI objects for System) and Context Entity Categories, Quality Focuses (just for EQ and QinU), Measurable Properties, Measurement and Evaluation

Taking into account the *influences* and *depends on* relationships between quality views, Fig. 3 shows that the QinU focus (System-in-Use Quality view) *depends on* the EQ focus (System Quality view). Considering some empirical observations made in [13], 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, Fig. 3 depicts two Context boxes, and Fig. 1 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 [17] 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, *ContextProperties* (Fig. 1) are used which are also *Attributes*.

3.2 Integrated Strategies for Measurement, Evaluation and Change

As above mentioned, ME/MEC strategies are the second pillar of our *holistic quality evaluation 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 project goal. An integrated strategy should support simultaneously three principles or capabilities [4]: a *domain conceptual base and framework*, *process perspective specifications*, and *method specifications*.

The first principle is the C-INCAMI conceptual base (Fig. 1), 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. Besides, process specifications can consider different process perspectives such as functional, behavioral, informational and organizational [6]. Process specifications should primarily state what to do rather than indicate the particular methods and tools (resources) used by specific activity descriptions. The third capability allows assigning systematically particular ways (methods) to ultimately perform the ME and MEC tasks.

In the last decade, we developed specific strategies such as GOCAME and SIQinU, which have been employed in concrete evaluation and improvement projects. GOCAME embraces one quality view, and SIQinU two quality views. For instance, SIQinU supports the QinU/EQ/QinU evaluation and improvement cycles, starting evaluations from Task-based and/or Perception-based App-in-use entities, and their corresponding characteristics and attributes. Thus, we have employed SIQinU in the JIRA case study [13] for the most frequently used "Entering a new defect" task, and its associated Actual Usability sub-characteristics and attributes. Besides, for the Perception-based App-in-use sub-entity we have evaluated Satisfaction sub-characteristics and attributes by means of questionnaire items. Ultimately, SIQinU relates just the System-in-Use Quality view with the System Quality view exploring the abovementioned relationships between views.

To summarize, our *holistic quality evaluation approach* was used by means of quality views and specific strategies. However, we have recently observed that a ME or MEC strategy can be applied to recurrent problems of any ME/MEC project. Thus, we sought to develop a set of strategy patterns that offer reusable and instantiable solutions. Patterns are essentially ‘experience in a can’, for our case, ready to be opened and used by evaluators.

4 Documenting the GOCAMEC_1V Strategy Pattern

In software engineering, a well-known reference in defining design patterns is [8] that says “*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*”. 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 is to provide a reusable and customized solution to a recurrent problem in similar situations.

Let's suppose that the project objective is to "improve" one of the three quality views colored in Fig 2. This objective usually implies to understand the current state of an entity, and then understand its further state, after performed changes. For the same "improve" objective, even though there could exist different quality views (e.g., Product, System, and System-in-Use Quality views), the same strategy pattern can be used. Hence, for a concrete MEC project for one quality view, the same MEC strategy pattern should be customized accordingly. That is, its MEC processes and methods should be instantiated appropriately for the intended view. Thus, we describe next the *Goal-Oriented Context-Aware Measurement, Evaluation and Change for one Quality View* (GOCAMEC_1V) strategy pattern which can be applied to MEC projects.

We have specified a set of strategy patterns, following to some extent the pattern specification template used in [8]. Our template includes the following items: (1) *name*: A descriptive and unique name, usually expressed in English; (2) *alias*: Acronym or other names for the pattern; (3) *intent*: Main objective for the pattern; (4) *motivation* (*problem*): Problem which solves the pattern; (5) *applicability*: Situations in which the pattern can be applied; (6) *structure* (*solution*): Generic structure and instantiable solution that the pattern offers; (7) *known uses*: References of real usage; (8) *scenario of use*: Concrete example and illustration for the instantiated pattern. Below, these items are used to specify the GOCAMEC_1V pattern:

Name: *Goal-Oriented Context-Aware Measurement, Evaluation and Change for one Quality View*. **Alias**: GOCAMEC_1V.

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 applied 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 (as shown in Fig. 2).

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. GOCAMEC_1V provides a generic course of action solution that indicates which activities must be instantiated during project planning. It also provides method specifications which indicate how the activities can 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 `requirements`, `context`, `measurement`, `evaluation`, `change`, and `analysis` components (Fig. 1) are defined as a domain ontology. The included terms, attributes and relationships belong to the MEC area.
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. 4. depicts only 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 1 exemplifies a method specification template for an indirect metric metadata, used in an indirect measurement task. Note that terms in method specification templates come from the MEC 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.

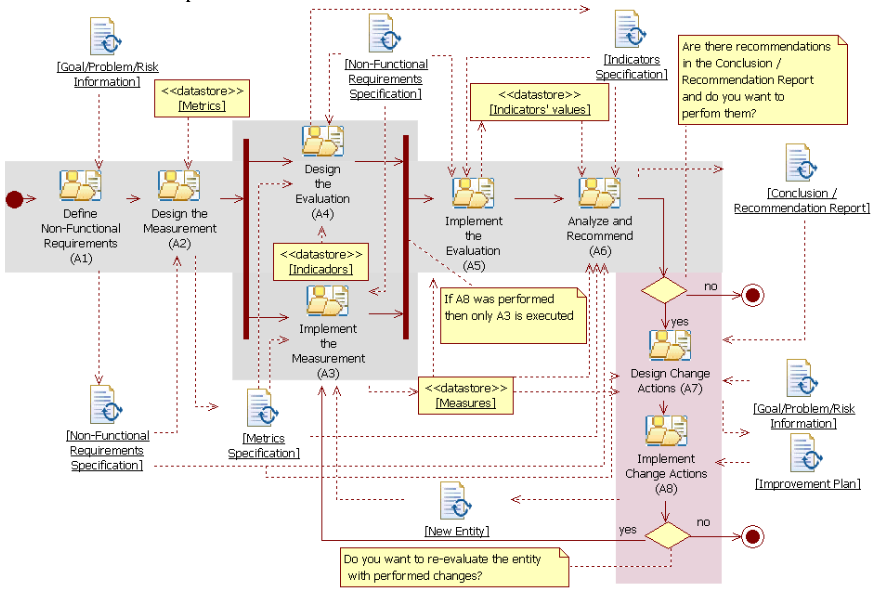
[Goal/Problem/Risk Information]
<<datastore>> [Metrics]
[Non-Functional Requirements Specification]
[Indicators Specification]
<<datastore>> [Indicators' values]
Are there recommendations in the Conclusion / Recommendation Report and do you want to perfom them?
Define Non-Functional Requirements (A1)
Design the Measurement (A2)
Design the Evaluation (A4)
<<datastore>> [Indicadors]
Implement the Measurement (A3)
Implement the Evaluation (A5)
Analyze and Recommend (A6)
[Conclusion / Recommendation Report]
no
yes
If A8 was performed then only A3 is executed
Design Change Actions (A7)
[Goal/Problem/Risk Information]
<<datastore>> [Measures]
[Non-Functional Requirements Specification]
[Metrics Specification]
Implement Change Actions (A8)
[Improvement Plan]
[New Entity]
yes
no
Do you want to re-evaluate the entity with performed changes?

Fig. 4. Functional and behavioral generic process perspectives for the GOCAMEC_1V pattern

Table 1. Method specification template for an Indirect Metric

Indirect Metric		
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: Info about the used tool if any)		**Related Direct Metrics**

<u>Known uses</u>: GOCAMEC_1V was used for instance 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 [20].

<u>Scenario of use</u>: The present example stems from the Facebook's mobileapp Usability case study (using v3.8 for Android) made in Dec. 2013 [17], and then replicated in Sept. 2014, using v14, and applying 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 *QualityView* is "System Quality", i.e., the *Entity SuperCategory* is "System" and the concrete *Entity* is "Facebook mobile app". The *QualityFocus* is "EQ" where the evaluated characteristic (*CalculableConcept*) is "Usability" and its related "Learnability", "Operability", etc., sub-characteristics (refer to [17], Table 1, for full definitions of the used Usability sub-characteristics and attributes).

Consequently, for the above project objective, it is suitable to instantiate the GOCAMEC_1V pattern. That is, the strategy pattern for one quality view and MEC purposes should be selected and instantiated for the "System Quality view" accordingly. Therefore, the 8 activities that the pattern provides as a solution (Fig. 4) 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.

The instantiated A1 activity produces the "Non-Functional Requirements Specification for EQ" document, which includes the "Information Need Specification for EQ" (Fig. 5), and the "Requirements Tree for EQ" (Table 2) artifacts. Table 2, 1st column, shows an excerpt for the Usability sub-characteristics and *Attributes*.

Information Need's purpose: Improve **User Viewpoint**: Final user
Entity Category: Social Network Application **Entity Super-Category:** System
(Concrete) Entity: Facebook mobile app, version 14 for Android
Quality Focus*:* External Quality
Context Properties: *Mobile device type*: "Mobilephone", *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. 5. Summarized *Information Need* artifact produced in the instantiated A1 activity

In the MEC project's scheduling phase, methods (as a metric) 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. 4) and their specifications are based on templates such as one shown in Table 1. For instance, the "Ratio of Main Controls Permanence" (%MCP) *IndirectMetric* quantifies the "Permanence of main controls" attribute (coded as 1.3.3.1.1 in Table 2). 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*" (refer to [17], Table 3, for full details on the metric specification). Other suitable methods for the *Elementary* and *Derived Evaluation* tasks should be assigned in the instantiated A4 activity. For the former task, an *ElementaryIndicator* per each measured attribute should be selected. For the latter an aggregation scoring model to calculate and interpret Usability and its sub-characteristics should be assigned as well.

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 quoted *IndirectMetric* specification for quantifying the 1.3.3.1.1 attribute, it allows to record its availability per each "Main button" of the "Main controls bar" for each Facebook screen in which must remain permanent.

Table 2. Excerpt of the *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**🟢
1.2 Learnability		**59.7**🔴		**66.4**🟡
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🟢
1.3.3 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**🔴
1.5 UI Aesthetics		**80.8**🟢		**91.1**🟢

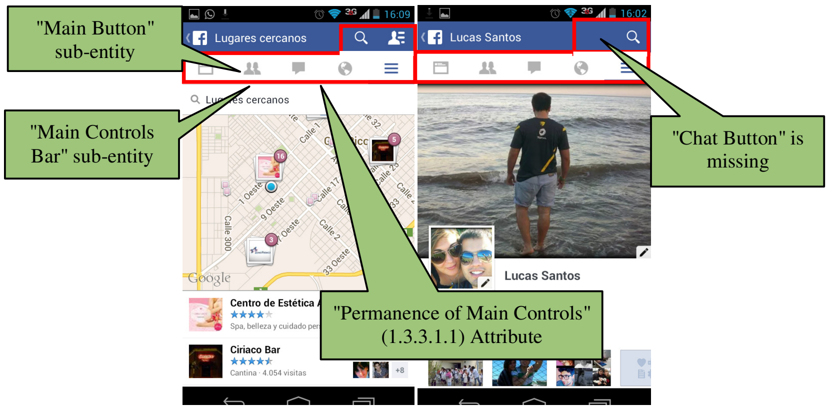

Fig. 6. Two Facebook screenshots. Left one shows the "Main Controls Bar" sub-entity, which is composed of seven "Buttons". Right one highlights the missing "Chat button"

Fig. 6 highlights in the left screenshot these two sub-entities and the quoted attribute. 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 %, however, all intermediate values for related direct metrics were recorded as well in the Measures <<datastore>> (see Fig. 4).

From the measures obtained in A3 and using the selected *ElementaryIndicators* and *DerivedIndicators* designed in A4, A5 must be instantiated (renamed as “Implement Evaluation for EQ”), and then executed. This activity produces the *Elementary* and *Derived Indicators*’ values shown in Table 2, 4th and 5th columns, for the re-evaluation performed in Sept. 2014. Note that in Table 2, the metaphor of the three-colored semaphore is used to identify the acceptability level of satisfaction met 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.

After finishing ME activities, the instantiated A6 ("Analyze and Recommend for EQ") activity should be performed. Table 3 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. A recommendation then follows to design and perform change actions for improvements. Each recommendation has also a priority. E.g., in Table 3,"H" means high priority.

Looking at the elementary indicator values for 2014 in Table 2, three attributes fell in the "Unsatisfactory" acceptability level, namely: "Mandatory entry" (1.3.1.2), "Text size appropriateness" (1.3.2.2.2), and "Permanence of main controls" (1.3.3.1.1). So, at least, three recommendations must be made with high priority for designing change actions. For instance, for the 1.3.3.1.1 attribute the R1.1 recommendation in Table 3 establishes that it is worth to “*ensure that in the set of selected screens, the main controls bar has always the same main buttons*”. Fig. 6 shows two selected screenshots belonging to the set of evaluated appropriate screens (34 out of 38) for the Facebook mobile app, version 14 for Android. The "Main controls bar" sub-entity has seven "Main buttons". (Recall that in the evaluation made in 2013, using version 3.8, there were five buttons in the "Main controls bar"). Also, it can be observed in the right screen that the specific "Chat button" is missing.

The next A7 activity specified in GOCAMEC_1V should be instantiated. So, A7 is renamed now as “Design Change Actions for EQ”. Table 4 shows a fragment of the "Improvement Plan for EQ" artifact. Basically, per each (R) recommendation in Table 3, the change actions (CA) planned, 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 CA1.1 in Table 4, derived from R1.1 in Table 3. It indicates: “*add those missing main buttons in the main control bar per each screen with the problem detected*”. To this end, the "GUI refactoring" change method can be used (as in [20]). 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. It is worthy to remark that due to the fact that Facebook is a proprietary system, changes actually couldn’t be performed since we didn’t have access to its source code and GUI objects.

Table 3. Fragment of the *Recommendation Report for EQ* artifact generated in the instantiated A6 activity. (Note: **H** means High, i.e., an urgent action is recommended)

ID	Recommendation (R)	Attribute	Priority
R1	**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

Table 4. Excerpt of the *Improvement Plan for EQ* produced in the instantiated A7 activity

ID	Change Action (CA)	CA sources	Method
CA1	**1.** Add those missing main buttons in the main control 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

Once changes are made through the instantiation of A8, the GOCAMEC_1V 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. 4. So the achieved improvement gain can be compared with the previous version. Thus, once the CA1.1 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

Using our holistic approach, *ME/MEC strategies* are chosen for evaluation and improvement embracing one or more *quality views* to accomplish the project goal. To this end, we have also envisioned to design a set of strategy patterns as a way of packaging general and reusable solutions for common, recurrent problems/goals within measurement, evaluation and change/improvement situations for specific projects. Hence, a strategy pattern, according to the project goal and the amount of involved quality views can be selected from the set of strategy patterns.

For example, if the project aims at assessing just one *quality view*, i.e., to understand the current situation of an entity category with regard to the corresponding quality focus, then the GOCAME_1V strategy pattern is the suitable. This pattern was conceived from the experience gained in concrete projects using the previously built GOCAME strategy. The pattern intention is to provide a solution in the instantiation of a ME strategy aimed at supporting an understanding project goal when one quality view is considered. The GOCAME_1V generic process consists of six activities, which are the gray-colored A1-A6 in Fig. 4. This is the simplest pattern to be instantiated and, so far, the mostly used in our ME projects: E.g., in the evaluation of a mashup application [18], a shopping cart [19], etc. Note that this pattern can be used to evaluate either IQ, or EQ, or QinU focuses considering Usability concepts. But this pattern is applicable to other quality views and focuses as well.

On one hand, if the project involves MEC goals for one *quality view* then the GOCAMEC_1V strategy pattern, as documented in Section 4, should be selected. We illustrated the Usability characteristic (linked to the EQ focus) for improving the Facebook social network app. We may also instantiate this pattern to improve for instance the Usability (IQ focus) of an artifact at early stages of development, such as an architectural design, as presented in [7]. On the other hand, if the project involves MEC goals for two *quality views* then the GOCAMEC_2V strategy pattern should be chosen. Recall that in Fig. 2, between two quality views, the *depends on* and *influences* relationships can be used. If we consider for a while, the System Quality and the System-in-Use Quality views, these relations embrace the hypothesis [12] that evaluating and improving the EQ of a system is one means of improving the QinU of a system in use. Similarly, evaluating the QinU can provide feedback to improve the EQ. Thus, the GOCAMEC_2V strategy pattern embeds this hypothesis. A concrete strategy derived from this pattern is the so-called SIQinU –used in an industrial case [13]-, which supports the QinU/EQ/QinU improvement cycles as commented in sub-section 3.2. Note that the GOCAMEC_2V strategy pattern can be instantiated to other 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.

Finally, it is worthy to remark that the inclusion of *strategy patterns* to our *holistic quality evaluation approach* makes it scalable. This is due to that the second pillar of our approach, thanks to the conception of different strategy patterns applied appropriately to quality views, makes a scalable approach considering the amount of quality views that a ME/MEC project can embrace. Furthermore, thanks to the first pillar, the modeling of quality multi-views and their relationships foster developing a family of patterns. Note that more quality views than those depicted in Fig. 2 can be derived from Fig. 1. That is, we can represent that the Process Quality view influences the Service Quality view, and in turn the latter to the Service-in-Use Quality view.

6 Conclusions and Future Work

As to the contributions of this work, we have enhanced our *holistic quality evaluation approach* by strengthening its architecture based on two pillars. Firstly, for the *quality multi-view modeling framework*, a *quality_view* component was added to the previously developed C-INCAMI conceptual framework. In this component the *QualityView* concept was modeled as an association between *QualityFocus* and *EntitySuperCategory* terms. Also, the *influences* and *depends on* relationships for views is explicitly represented. This component allows by instantiation a clear separation of concern with regard to entities/focuses and linked quality models (characteristics and attributes), as discussed in Section 3.

Secondly, due to the lessons learnt in the development of concrete ME/MEC strategies, we envisioned the opportunity to generalize the gained knowledge into strategy patterns. The benefits of having documented patterns is well known. Hence, we 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 relationships, in conjunction with project goals are key aspects to defining ME/MEC strategy patterns. Finally, we have illustrated the GOCAMEC_1V strategy pattern for the Facebook's mobileapp Usability case study, and analyzed the applicability of other strategy patterns in the light of scalability issues.

As a future line of research, we will instantiate the GOCAMEC_3V strategy pattern, for Usability and Security issues and architectural design artifacts for IQ, in order to gauge improvements in QinU for both Satisfaction and Usability-in-use characteristics, taking into account to a some extent the Folmer *et al.* work [7]. We think that by using our approach and a couple of suitable strategy patterns, many of the raised issues in [7] can be covered appropriately.

References

1. Alexander C: The Timeless Way of Building. Oxford University Press, (1979)
2. Barišić A., Monteiro P., Amaral V., Goulão M., Monteiro M.: Patterns for Evaluating

Usability of Domain-Specific Languages. In 19th Conference on Pattern Languages of Programs (PLoP) Tucson, USA, (2012)

3. 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)
4. 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), Paper 2, pp. 1-26. ISSN 0717-5000, (2015)
5. 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)
6. Curtis B., Kellner M., Over J.: Process Modelling. Com. of ACM, 35:(9), pp.75-90 (1992)
7. 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)
8. Gamma E., Helm R., Johnson R.,Vlissides J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addisson-Wesley, ISBN 0-201-63361-2, (1995)
9. Georgiakakis P., Retalis S., Psaromiligkos Y., Papadimitriou G.:. Depth toolkit: a web-based tool for designing and executing usability evaluations of e-sites based on design patterns. In Springer LNCS 4550, HCI'07, Jacko J. (Ed.), pp. 453-462, (2007)
10. 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)
11. 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)
12. ISO/IEC 25010: Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, (2011)
13. 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, Vol.17, No. 4, pp. 299-330, (2012)
14. Lindvall M., Donzelli P., Asgari S., Basili V.: Towards Reusable Measurement Patterns., 11th IEEE Int'l Symposium in Software Metrics, pp. 1-8, (2005)
15. 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)
16. Nielsen, J., Budiu, R.: Mobile Usability, New Riders, Berkeley CA, (2012)
17. 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)
18. 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)
19. 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)
20. 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)