Journal of Web Engineering, Vol. 7, No.XXX (2008) 000-000

WEB APPLICATION EVALUATION AND REFACTORING: A QUALITY-ORIENTED IMPROVEMENT APPROACH

LUIS OLSINA
GIDIS_Web, Engineering School at Universidad Nacional de La Pampa, Argentina.
olsinal@ing.unlpam.edu.ar

ALEJANDRA GARRIDO, GUSTAVO ROSSI
LIFIA, Universidad Nacional de La Plata and CONICET, Argentina.
{garrido, gustavo}@lifia.info.unlp.edu.ar

DAMIANO DISTANTE, GERARDO CANFORA
RCOST, University of Sannio, Italy.
{distante, canfora}@unisannio.it

Web applications must be usable and accessible; at the same time, their continuous evolution makes it difficult to keep a high degree of external quality. Refactoring is a practice of agile methods well-suited for the maintenance and evolution of Web applications. However, this practice is mainly intended and used to improve maintainability and extensibility of the design and code rather than external qualities such as usability. We believe that the concept of refactoring as "behavior-preserving transformations" can be applied to the navigation and presentation models of a Web application with the purpose of improving external quality. For this reason we have defined the concept of Web model refactoring. This paper demonstrates how it is possible to improve the external quality of a Web application by combining a mature quality measurement and evaluation method (WebQEM) with Web model refactoring. WebQEM is used to identify needs for improvement, recommend Web model refactorings and assess their impact on some defined attributes of a Web product entity. We present a case study showing how a typical shopping cart in an e-commerce site can improve its usability and content quality with our integrated improvement approach.

Key words: Web Model Refactoring, Usability Improvement, External Quality, Web Quality Evaluation, Metrics, Indicators, WebQEM, INCAMI.

1 Introduction

It is well known that Web applications evolve constantly following unpredictable patterns: requirements change, user feedback drives new functionality, new technologies enable new interaction paradigms, etc. Fortunately, many software engineering approaches, particularly agile methods, have emerged to help developers cope with, and even welcome, constant change in requirements [1]. Agile methods use short development iterations driven by tests, and encourage simple designs to solve the problem at hand. Refactoring is one of the fundamental practices of agile development used to add flexibility and extensibility before introducing new functionality [7].

Refactoring was defined in the context of object-oriented software to "factor out" new abstractions and apply other small transformations to the source code of an application without changing its behavior [25]. These transformations aim at improving internal qualities of the code, making it more flexible to subsequent semantic changes. Model refactoring has also been proposed to improve the

structure of the design, increasing reusability and maintainability while preserving behavior [3]. In turn, refactoring to patterns helps developers introduce patterns just when they are needed, thus keeping the balance between under-engineering (repeatedly adding new functionality without a previous clean-up) and over-engineering (applying design patterns to create overly complex designs) [15].

In the context of Web applications, we propose model refactorings for the navigation and presentation models, defining them as Web Model Refactorings (WMRs) [10]. As we describe later in the paper, the difference between WMR and conventional model refactoring is in the purpose, the subject of refactoring and the refactoring primitives. WMR aims at improving the application's usability instead of its internal quality. For example, WMRs may improve the application's navigational topology, and/or interface look and feel and may guide the introduction of Web patterns [33] into the application's structure. Regarding the subject of refactoring, WMRs are applied on the navigation and presentation models of a Web application, while conventional model refactoring is applied to domain models (usually UML class diagrams). Concerning the refactoring primitives, they are also different since conventional model refactorings restructure artifacts of UML diagrams (extracting methods, abstracting super-classes, replacing inheritance with delegation, etc. [7]), while WMRs apply transformations on design primitives of the navigation and presentation models of a Web application (adding attributes to navigation nodes, anticipating the target of a link, introducing scrolling, splitting a page, etc. [10]).

In this paper we present how to combine WMR with a quality evaluation method; using a well-established measurement and evaluation approach [19, 22] allows guiding the refactoring process and assessing the improvement of a Web application's quality features after refactoring.

We next present an example to illustrate our ideas. Figure 1.a shows a reduced version of the Amazon shopping cart, and Figure 1.b shows the same cart with some added information and operations. In our research we want to be able to (*i*) identify needs / opportunities for refactoring of the navigation and presentation models, (*ii*) specify the kind of transformations that may lead to improvements like those illustrated in the example, and at the same time, (*iii*) be able to measure and evaluate the quality improvements actually obtained.

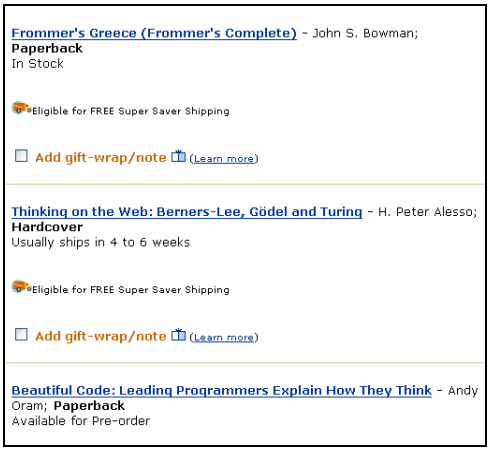

Figure 1.a: Basic shopping cart

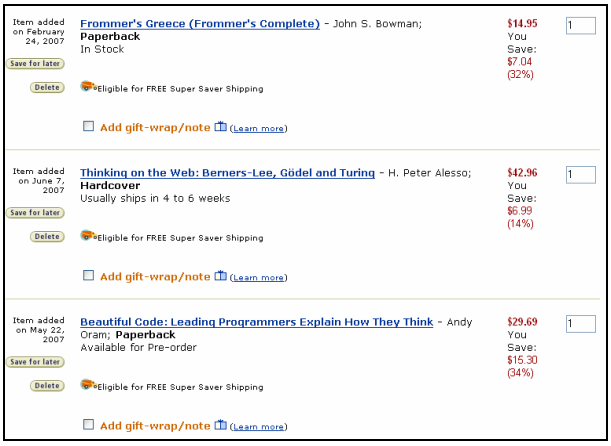

Figure 1.b: Enhanced shopping cart

By explaining the integrated approach thoroughly, this paper extends the work presented in [18] and provides the following main contributions:

- We propose WMRs as a way to incrementally improve the external, user perceivable, qualities of a Web application;
- We show how to integrate quality evaluation in the process of refactoring to identify needs for refactoring and assess the actual quality improvements after refactoring;
- We demonstrate how WMRs actually improve usability and content quality on a particular case study.

The rest of the paper is structured as follows. Section 2 provides a summary of our quality-oriented improvement process. Section 3 gives an overview of a measurement and evaluation approach and its underlying quality evaluation method (WebQEM). Section 4 presents the concept of WMR. Section 5 shows how we combine WebQEM and WMR to improve the design of the shopping cart of an e-commerce site. Section 6 discusses some related work and Section 7 concludes the paper and describes some further research we are pursuing.

2 Overview of the Quality-Oriented Improvement Approach

In [18] we have proposed the use of WMR to improve the usability of a Web application and the use of WebQEM to measure and evaluate usability before and after refactoring, thus demonstrating the improvement gained by WMR. In this paper we extend this idea, describing the whole process of quality evaluation with WebQEM and the details behind WMR, in order to reach a closer integration of both approaches. The integrated process is composed of six (6) activities. The first three activities should be performed only once during the application development lifecycle. The other three activities are performed iteratively and more frequently; how often depends on how agile is the development process followed. The six main activities are:

1. Definition of the quality attributes expected for the Web application for a given information need. In the context of an agile methodology, this activity is performed by experts with the customer or final user of the application at the beginning of the lifecycle. Note that attributes can be selected from a catalogue [17].
2. For each attribute defined in 1, the design (or selection from a catalogue) of a metric and the execution of the measurement.
3. Design (or selection) of elementary indicators that represent the satisfaction level of each attribute, and execution of the evaluation.
4. Analysis of the evaluation results and drawing of recommendations for WMRs that may improve each attribute with an unsatisfactory level, if any. Note that the mapping between WMRs and attributes may also exist in a catalogue.
5. Application of the recommended WMRs, keeping track of the attributes affected by the changes.
6. Execution of the measurement and evaluation, and analysis of the results on only those attributes changed in 5. Go back to 4.

3 Overview of the Quality Measurement and Evaluation Approach

Developing and maintaining successful Web applications with economic and quality issues in mind requires the effective incorporation of a number of principles, models and methods. Particularly a common challenge faced by many Web development organizations is to have a clear establishment of measurement and evaluation approaches for quality assurance projects and programs. We argue that at least three pillars are necessary to build a sound measurement and evaluation approach:

1. A process for measurement and evaluation, i.e. the main managerial and technical activities that might be planned and performed.
2. A measurement and evaluation framework that must rely on a sound conceptual base.
3. Specific model-based methods and techniques in order to carry out specific activities.

First, a *measurement and evaluation process* prescribes a set of main phases, activities and their inputs and outputs that might be considered [19]. Usually, it states what to do but not how to do it. An overview of this process is given in subsection 3.1.

Second, a sound *measurement and evaluation conceptual framework* is necessary. Often organizations start measurement programs from scratch more than once because they do not pay attention to the way nonfunctional requirements, metrics and indicators should be designed, recorded and analyzed. A well-established framework has to be built on a sound conceptual base, i.e., on an ontological base, which explicitly and formally specifies the main agreed concepts, properties, relationships, and constraints for a given domain. In this direction, we have built an explicit specification for the metrics and indicators domain [21]. However, the metrics and indicators ontology itself is not sufficient to model a full-fledged measurement and evaluation framework, but rather the rationale for building it. In [19], the INCAMI (*Information Need*, *Concept model*, *Attribute*, *Metric* and *Indicator*) framework is thoroughly analyzed in the light of its ontological roots and its five main components. An overview of this framework is given in subsection 3.2.

Finally, the third tenet of our approach is the *method*. As aforementioned, while a measurement and evaluation process specifies what to do, a method specifies how to perform the prescribed activities relying on specific models and criteria. In subsections 3.3 and 5.2 we illustrate from the practical point of view the Web Quality Evaluation Method (WebQEM) [22], which is an instantiation of INCAMI.

3.1 The Measurement and Evaluation Process

The underlying WebQEM process integrates activities for nonfunctional requirements, measurement, evaluation, analysis and recommendations [19]. Figure 2 shows the evaluation process including the main technical processes (represented as boxes), inputs and outputs (represented with arrows). This proposal follows to some extent the ISO's process model for evaluators [12], and the software measurement process [14]. It comprises eight activities that we describe below.

1. Nonfunctional Requirement Definition. This activity has as inputs the specific *information needs* (basically the purpose for the evaluation and user viewpoint); the descriptions of the *entity*, e.g. the Amazon shopping cart, as a sub-product of the whole Web application, etc., and potential quality models, e.g. as those in ISO 9126-1 [13]. The output of this process is a nonfunctional requirement specification document containing for instance an instantiated quality model, i.e. a *requirement tree*

relating characteristics (*calculable concepts* as usability, accessibility, etc.), sub-characteristics (or sub-concepts like operability, etc.), and associated *attributes* as atomic, measurable properties of an entity. (The reader can refer to Table 3 in the Appendix for the definition of the terms highlighted in this subsection in italic).

2. Measurement Design. Its input is basically the attributes from the requirement tree and its output is the designed metrics. One *metric* should be designed (or selected from a catalogue) per each attribute. The output (the metric specification) contains the *scale*, *scale type*, *measurement method*, *unit* (for numerical scales), among other metadata.

3. Elementary Evaluation Design. It has as inputs the attributes from the requirement tree and their metrics, the *elementary model* and the *decision criteria*. One *elementary indicator* should be designed (or selected from a catalogue) per each elementary requirement (attribute) of the tree, and will represent in the end its satisfaction level. The output (the elementary indicator specification) contains the *scale* and *acceptability levels* among other metadata.

4. Global Evaluation Design. Its input is the calculable concepts and sub-concepts from the requirement tree and the elementary indicators, as well as the *aggregation model* and the decision criteria. There are different aggregation models for evaluation purposes such as linear additive or nonlinear multi-criteria scoring models, among others. The output is the global indicator specification, useful to calculate the requirement tree and to determine ultimately the partial and global level of satisfaction achieved for the given information need.

5. Measurement Implementation (or Execution). It has as input basically the designed metric per each attribute, the instance of the entity, the tool (if any) for data collection that automates the metric's measurement method (this input is not shown in Figure 2), and as output the *measure* value. A *measurement* execution process is made at a given time point and therefore a collection of values from the same measurement at different time periods conform a measure set.

6. Elementary Evaluation Implementation. Its input is the metric's value, the elementary indicator specification, the tool (if any) for calculation, and its output is the yielded elementary *indicator value*.

7. Global Evaluation Implementation. It has as input the elementary indicator values, the global indicator specification. Its output is the yielded partial/global indicator value.

8. Analysis and Recommendation. Its input is composed of the entity descriptions and instance/s, the specification of nonfunctional requirements, the metrics and indicators specifications, as well as the measures (data) and elementary, partial and global indicator values (information). Requesters and evaluators can then analyze and understand for example the assessed Web product's strengths and weaknesses with regard to established information needs, and suggest and justify recommendations. The output (not shown in the figure) can be an analysis report and/or a recommendation report.

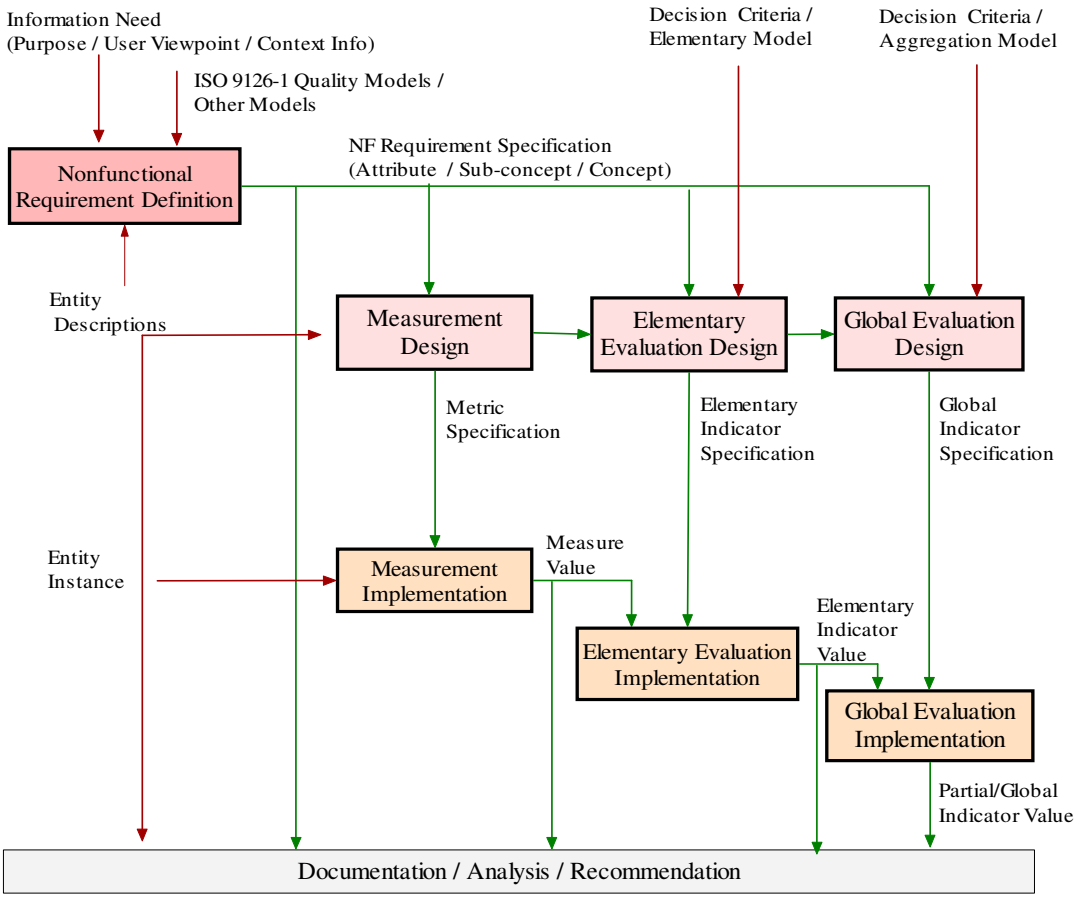

Figure 2: The basic measurement and evaluation process underlying the WebQEM methodology.

3.2 The INCAMI Framework

The INCAMI [19] framework is based upon the assumption that an organization seeking to measure and evaluate a project in a purpose-oriented way, must first specify nonfunctional requirements starting from information needs. Next, the organization must design and select the specific set of useful metrics for measurement purpose. Lastly, it must interpret the obtained values by means of contextual indicators, evaluate the degree in which the stated requirements have been met, and ultimately, draw conclusions and give recommendations. As aforementioned, the conceptual framework is made up of five main components, namely:

1. Measurement and Evaluation Project Definition.
2. Nonfunctional Requirement Definition and Specification
3. Measurement Design and Execution
4. Evaluation Design and Execution
5. Analysis and Recommendation

Most of the components are supported by many of the ontological terms defined in previous works [17, 19, 21]. In Figure 3, we show an abridged UML diagram in which the main terms, attributes and relationships for the components 2, 3 and 4 are integrated. Next, we highlight the text with italic for those terms and attributes that appear in the diagram - also the reader can refer to Table 3 in the Appendix for the definition of the terms only, which are grouped by component accordingly).

In order to meet the Nonfunctional Requirement Definition and Specification component, the *Information Need* for a measurement and evaluation project must be agreed upon. Usually, information needs come from goals that decision-makers seek to achieve. The *InformationNeed* class (Figure 3) has three properties, viz. the *purpose*, the user *viewpoint*, and the *contextInformation*. Besides, the *InformationNeed* class has two main relationships with the *CalcuableConcept* and the *EntityCategory* classes respectively. A calculable concept can be defined as an abstract relationship between attributes of entities' categories and information needs; in fact, internal quality, external quality, cost, etc. are instances of a calculable concept. In turn, a calculable concept can be represented by a *ConceptModel*, like ISO 9126-1 specifies software quality models. On the other hand, a common practice is to assess quality by means of the quantification of lower abstraction concepts such as *Attributes* of entities' categories. The attribute term can be shortly defined as a measurable property of an *EntityCategory* (e.g. categories of entities of interest to Software and Web Engineering are resource, process, product, service, and project as a whole). In summary, this component allows the definition, specification and instantiation of nonfunctional requirements in a sound and well-established way.

Regarding the Measurement Design and Execution component, purposeful metrics should be selected in the process. In general, each attribute can be quantified by many metrics, but in practice just one metric should be selected for each attribute, given a specific measurement project. The *Metric* concept contains the definition of the selected *Measurement* or Calculation *Method* and the *Scale*. We can apply an objective or subjective measurement method for *Direct Metrics*; conversely, we can perform a calculation method for *Indirect Metrics*, that is, when a *formula* intervene. Once the metric has been selected, we can perform the measurement process, i.e., the activity that uses a metric definition in order to produce a *measure*'s value. Class *Measurement* allows recording the date/*time stamp*, the information of the owner in charge of the measurement activity, and the actual or estimated yielded *value*. However, since the value of a particular metric will not represent the elementary requirement's satisfaction level, we need to define a new mapping that will produce an elementary indicator value. One fact worth mentioning is that the selected metrics are useful for a measurement process as long as the selected indicators are useful for an evaluation process in order to interpret the stated information need.

For the Evaluation Design and Execution component, contextual indicators should be selected. Indicators are ultimately the foundation for interpretation of information needs and decision-making. There are two types of indicators: *Elementary* and *Global Indicators*. Particularly, we define an elementary indicator as one which does not depend upon other indicators to evaluate or estimate a concept at a lower level of abstraction (i.e., for associated attributes to a concept model). On the other side, we define a partial or global indicator as one which is derived from other indicators to evaluate or estimate a concept at a higher level of abstraction (i.e., for sub-concepts and concepts).

The global indicator's value ultimately represents the global degree of satisfaction in meeting the stated requirements (information need) for a given purpose and user viewpoint.

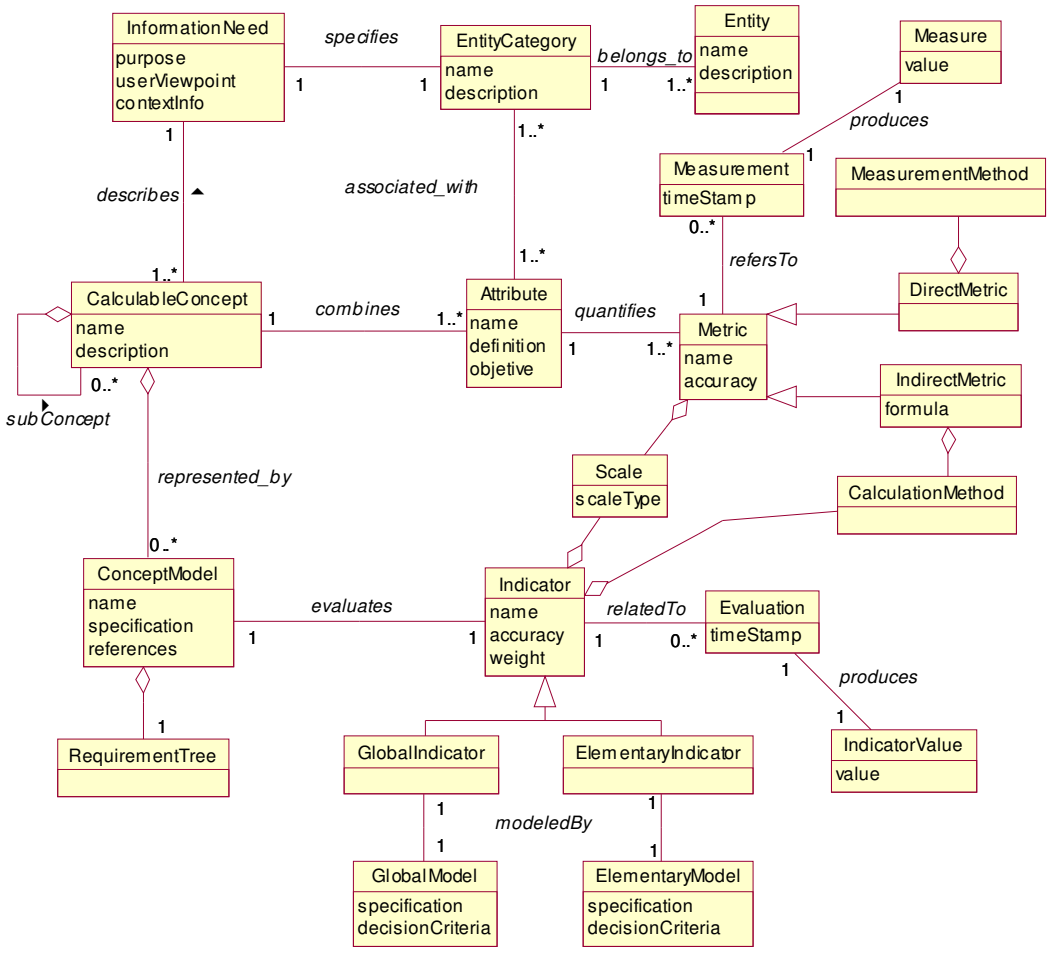

Figure 3: Key terms and relationships in the non-functional requirement, measurement and evaluation components.

3.3 The WebQEM Methodology

WebQEM [22] is a purpose-oriented evaluation methodology for the inspection of calculable concepts, sub-concepts, and attributes stemming from an instantiated quality model. WebQEM relies on the process outlined in section 2.1, and on the conceptual framework previously presented. We have been developing the WebQEM methodology and its associated tool since the late 1990s. It has been used to evaluate Web applications in several domains as documented elsewhere [6, 23, 24]. We present WebQEM and its applicability to refactoring in Section 5.2.

4 Overview of Web Model Refactoring

4.1 Model Refactoring

Refactoring was originally conceived as a technique that applies syntactic transformations to the source code of an application without changing its behavior but improving its readability, extensibility and reusability [25]. Web code refactorings, e.g. those applied to the source code or HTML structures to improve code quality, are outside the scope of this paper but discussed elsewhere [27, 28].

In addition to assisting developers in improving code quality, refactoring techniques can also support the process of continuous improvement of an application's design. As Boger et al. first proposed in [3], a set of refactorings can be identified to improve the structure of software at the model level (UML class diagram), where the graphical representation provides a higher-level insight of the application's design, which is not apparent from the code.

Further research has been published in the area of model refactoring for other UML diagrams [34] and tools for model refactoring [35]. Beyond the identification of new refactorings at the model level, their ultimate purpose is always to improve internal qualities of the application.

4.2 Web Model Refactoring

The design of a Web application is usually accomplished in a three-stage process resulting in three models: *application (or content) model*, *navigation model* and *presentation model* [5, 16, 31, 32]. Model refactoring may be applied on each of these three models. Since the application model is a domain model independent of navigation or interface concepts, it is the subject of the same kind of model refactoring described above [3, 34, 35]. The navigation and presentation models of a Web application, instead, present new and special opportunities for model refactoring. We have identified and classified a set of navigation and presentation model refactorings that we generically call *Web Model Refactorings* (WMRs) [10].

4.2.1 Where are WMRs applied?

Navigation model refactorings apply small changes to the navigation model of a Web application. In general, this model defines the paths through contents and the operations that will be made available to the users, and is usually described with a navigational diagram composed of nodes, links, indexes and other access structures. Navigation model refactorings may thus change, among others: the contents available in a node, the set of outgoing links of a node, the navigation topology between a set of nodes, and the user operations accessible from a node [10].

Similarly, presentation model refactorings apply changes to the pages, widgets and interface controls of the presentation model. This model describes the look and feel of pages, the interface widgets they contain, the interface controls that trigger the application functionality and the interface transformations occurring as a result of user interaction. Presentation model refactorings may thus change the look and feel of a page by changing the type of widgets or the interface effects, by adding new widgets or anchors to enhance understanding, etc.

4.2.2 Why are WMRs applied?

Notice that navigation and presentation model refactorings apply over artifacts that the user perceives: navigation paths and interface specifications. They may affect the way in which the application presents contents, enables navigation through contents and provides interaction capabilities [10]. That is, WMRs transform the models applying changes that are reflected in the way the final user may interact with the Web application. Therefore, the intent of WMRs is to enhance the application's external quality, i.e., its usability.

WMRs focus on those (small) changes that may improve comprehension, facilitate navigation, smooth the progress of operations and business transactions, etc. Furthermore, in the same way as traditional refactorings may be used to introduce design patterns [15], Web model refactorings may be also driven by Web patterns [33] and therefore produce the same well-known benefits of the patterns they introduce.

4.2.3 How are WMRs applied?

WMRs follow the same spirit of traditional refactorings in the sense that they should preserve the behavior of a Web application. How is this behavior defined and how is it preserved is the subject of this subsection.

Behavior defined by the navigation model of a Web application is represented by: (*i*) the set of available nodes and navigation links between nodes; (*ii*) the set of available user operations and the semantics of each operation. Therefore, navigation model refactorings have to preserve the set of possible operations and their semantics, and the navigability of the set of nodes. Preserving the navigability of the set of nodes means that existing nodes may not become unreachable though the set may be augmented (e.g., by splitting a node). Moreover, these refactorings should not introduce information, relationships or operations that are not in the application model, thus preserving the application's functionality.

Behavior defined by the presentation model, which must be preserved by presentation model refactorings, is defined in terms of the actions that the user may trigger in a page, including both operations and link activation of the underlying nodes.

It is worth to note that most navigation model refactorings may imply some kind of refactoring at the presentation model (e.g., new information added to a node requires the corresponding user interface to change in order to present the additional information). In contrast, presentation model refactorings must be neutral to the underlying navigation structure.

4.3 Examples of WMRs

To illustrate the ideas presented above, we next describe some representative WMRs using a simplified template comprising the *type* of refactoring (navigation or presentation), the *motivation* for applying it, the refactoring *mechanics* and a practical *example* of its application. A more extensive list of WMRs, which include also composite WMRs and refactoring to RIA can be found in [10, 30].

4.3.1 Add Node Attribute (Navigation model refactoring)

Motivation: sometimes the designer may find the opportunity to provide the user with more information than that currently displayed on a certain page of the application. The information may come from different sources and have different purposes: it may be data extracted from the application

model, or it may be data introduced to help during navigation (obtained from the navigation model itself). This refactoring may be used to introduce patterns like *Clean Product Details* [33] to add details about products in an e-commerce website or *Personalized Recommendations* [33].

Mechanics: add an attribute to the node class in the navigation model that was identified in need of more information. If the information is extracted from the application model, attach to the attribute the statement describing the mapping to the application model.

Example: this kind of refactoring can be applied to the node underlying the page shown in Figure 1.a, to display information about price and savings of each product in the list. As a consequence, we obtain the enriched version of the shopping cart shown in Figure 1.b. It is worth noting that the information added to the shopping cart node is already available in the content model of the application.

4.3.2 Turn Information into Link (Navigation model refactoring)

***Motivation*:** during the process of completing a business transaction, some Web pages may show intermediate results or a succinct review of the information gathered until a certain point. This usually occurs, for example, when checking the status of the shopping cart during the process of buying some products in an e-commerce Web site. Such Web pages should provide the user with the chance to review the information associated to the intermediate results (e.g., items in the shopping cart) by means of direct links to the pages showing details on them. When this does not happen, the page can undergo the kind of refactoring we propose here for improvement.

Mechanics: select the portion of the information about the target item that better distinguishes it. In the navigational diagram, add a link to the corresponding item; the anchor of the link could be the selected portion of information.

Example: this refactoring may be used to add links from names of products in a shopping cart to the pages showing detailed information about the products (as we will illustrate later on).

4.3.3 Replace Widget (Presentation model refactoring)

Motivation: the presentation model describes, for each node in the navigation model, the kind of widgets that display each data attribute and those that permit to activate operations or links. Inspection of usage of the site may show that some information item or operation should be displayed with a different widget, to improve operability, usability or accessibility.

Mechanics: in the page of the presentation model that contains the widget found unsuitable, replace the current widget by a more appropriate one.

Example: check-boxes are best suited when items must be selected from a list to later perform an operation on them. A typical example is that of an email reader that allows selecting individual emails by means of check-boxes in order to apply, afterwards, operations like “delete” or “move to mailbox”. Consequently, users do not expect check-boxes to dispatch an operation when they are clicked but just to show a check-mark in the box. In the case of Cuspide’s shopping cart, which appears in Figure 4, checking any box under the title “Borrar” directly deletes the item from the shopping cart, which is usually confusing and does not allow changing one’s mind. In this case, a more suitable widget would be a button with label “Borrar” or the usual trash can icon (see Figure 5).

5 Evaluation and Refactoring: An Integrated Quality-Oriented Improvement Approach

Our proposal to continuously improve the external quality of a Web application, such as its usability, is to enrich the Web Engineering life cycle with a new activity: Web model refactoring (WMR). After each requirement/design/implementation cycle, and before the next cycle begins, the navigation and presentation models should be inspected in order to find opportunities for refactoring. Thus, the design quality will be incrementally improved, namely, the external quality and ultimately the quality in use.

5.1 The Integrated Process

The opening of this section raises the following issue: how do we find opportunities for refactoring? With conventional refactoring, Beck and Fowler suggest looking for what they call "bad smells" [7]: certain structures that experienced developers would find as sources of maintainability problems (e.g., a "Large class" or "Parallel inheritance hierarchies" [7]). This informal approach follows the agile philosophy of giving more value to human intuition and oral communication than to metrics and documentation [2, 7]. While this may be true for highly trained and skilled developers that follow most of the practices of Extreme Programming [2], we believe a more formal and systematic method of finding opportunities for refactoring can be insightful and enlightening on how to proceed. In particular, we propose using the WebQEM measurement and evaluation method to identify sources of usability problems that may profit from WMR.

Furthermore, we also propose using WebQEM afterwards, to assess the benefits of applying WMR. This allows evaluating the application's quality before and after the refactoring process, thus demonstrating the improvement gain. Moreover, when this process is systematically applied, the amount of measurement and evaluation needed after refactoring is reduced to the subjects of changes, which should be precisely described by the refactoring. In essence, our strategy shows a first step to systematize the process of WMR as a quality-driven activity.

The rest of this section describes our integrated approach by using a small component of a regional e-bookstore site, Cuspide (www.cuspide.com.ar). We analyze the shopping cart features of this site and concretely show how its usability and content quality improve after applying some simple WMRs. During our case study, we also show how each refactoring is mapped to specific quality attributes, which can contribute to the reuse and automation process.

5.2 The Shopping Cart Case Study

The external quality (particularly, aspects related to the usability and content) of the Cuspide shopping cart can be improved by systematically applying WMR and evaluations. Instead of using a set of "good practices" for shopping carts, we have used the Amazon.com shopping cart as a well-established reference for desirable requirements and quality attributes. In fact, Amazon is certainly a well-known instantiation of good practices for this type of application component.

In order to identify where WMR is needed and measure its impact on the usability of the application, we need to carry out the six activities outlined in Section 2: (*1*) specify the external quality attributes of the Cuspide shopping cart with regard to usability and content's quality; (*2*) consider

attributes with related metrics and (*3*) select indicators of the well-satisfied features of the Amazon shopping cart, carrying out the measurement and evaluation activities; (*4*) analyze the results and suggest which WMRs may apply; (*5*) follow the refactoring process and, finally, (*6*) measure and evaluate again to assess the benefits.

The first four (4) activities, which cover the definition of the external quality attributes, metrics and indicators, and their evaluation to recommend venue for improvements through WMR, are accomplished by applying the WebQEM method. Taken into account the measurement and evaluation background given in Section 3, in the first four subsections below we illustrate the main steps of WebQEM for the assessment of the Cuspide shopping cart entity, namely: the nonfunctional requirement definition, particularly the specification of two dimensions for the external quality model, i.e. the usability and the content quality (Subsection 5.2.1); the design and execution of the measurement (Subsection 5.2.2); the design and execution of the elementary and global evaluation (Subsection 5.2.3); and the analysis and recommendation step for the assessment before refactoring (Subsection 5.2.4). The last two subsections describe the application of three WMRs (Subsection 5.2.5) and the assessment of the final result (Subsection 5.2.6).

5.2.1 External Quality Specification

It is worth mentioning that quality is not an absolute concept but rather a relative and contextual one. Quality can be defined as an abstract relationship between attributes of an entity category (e.g., a product, a process, etc.) and a particular information need. Also there are different quality perspectives [20]; for instance, to software or Web applications we can specify, measure, and evaluate the following perspectives:

- *Internal Quality*, which is specified by a quality model (e.g., in [13] a set of six characteristics –Usability, Functionality, Reliability, Efficiency, Maintainability and Portability- and a set of sub-characteristics for each characteristic are prescribed), and can be measured and evaluated by static attributes of documents such as specification of requirements, architecture, or design; pieces of source code, and so forth. In early phases of a Web life cycle, we can evaluate, monitor and control the internal quality of these by-products, but assuring internal quality is not usually sufficient to assure external quality.
- *External Quality*, which is specified by a quality model (likewise as in the previous cited model), and can be measured and evaluated by dynamic properties of the running code in a computer system, i.e., when the module or full application is executed in a computer or network simulating as closely as possible the actual environment. In late lifecycle phases (e.g., in different kinds of testing, or even in the acceptance testing, or furthermore in the operational state of a Web application, as is in our case study), we can measure, evaluate and control the external quality of these late products, but assuring external quality is not usually sufficient to assure quality in use.
- *Quality in Use*, which is specified by a quality model (e.g., in [13] a set of four characteristics –Effectiveness, Productivity, Safety and Satisfaction- is prescribed), and can be measured and evaluated by the extent to which a software or Web application meets specific user needs in an actual, specific, context of use.

Each of these perspectives has its own added value considering a quality assurance strategy in the overall lifecycle. For example, assuring the external quality of a product can help evaluators to predict

the quality in use. A broader discussion on quality and its perspectives appears in [20]; an instantiation of the quality in use model for an e-learning case study is documented in [6]. In addition, the external quality model, a specific instantiation, as well as the justification for the inclusion of the *Content* characteristic and subcharacteristics for assessing the quality of information in the Web are illustrated in [20].

Many potential attributes, both generic and domain specific, can contribute to Web quality. However, as mentioned earlier, evaluation must be purpose-oriented for an actual information need. Referring back to Figure 3, we can state that given the *entity* instance (the Cuspide shopping cart), the *information need* can be specified by its *purpose* (suggest or assess WMRs), from the *user viewpoint* (final user), in a given *context* (using WMR in the context of an agile process). Moreover the information need has the focus on a *calculable concept* (external quality) and *subconcepts* (usability and content), which can be represented by a *concept model* (external quality model) and associated *attributes*.

Figure 4 shows a snapshot of the *entity instance* to be assessed, whose *entity name* is "Cuspide shopping cart", which in turn belongs to the "Web product" *entity category name*. The requirement tree specification including the *Usability* (1) and *Content* (2) characteristics, some subcharacteristics, and sixteen attributes related to them appear in the left column of Table 1. For instance, the Usability characteristic splits into subconcepts such as *Understandability* (1.1), *Learnability* (1.2), and *Operability* (1.3).

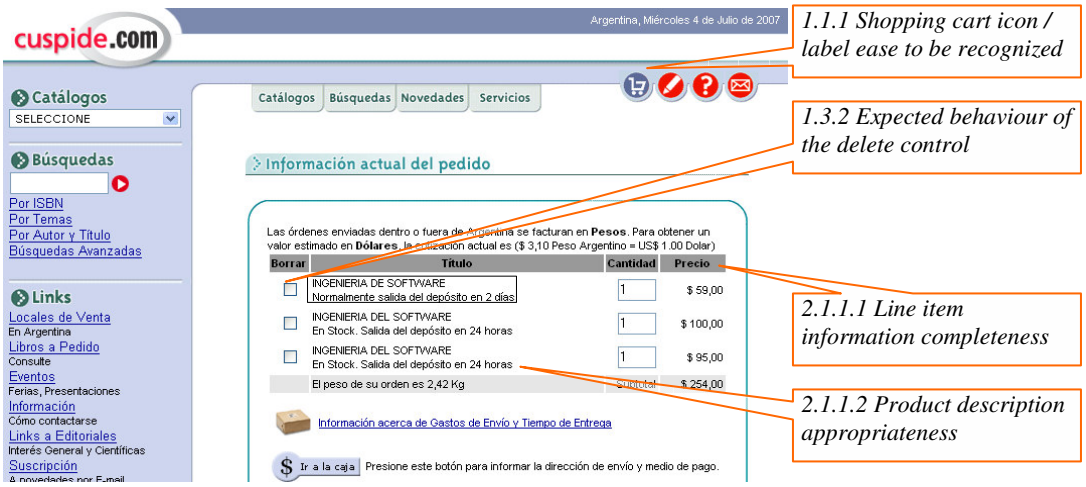

Figure 4: Screenshot of the Cuspide's shopping cart (before refactoring) with four attributes highlighted.

For this study, in the Content dimension we just consider the *Information Suitability* (2.1) subconcept, which we define as "the capability of a web product to deliver contents with the right coverage and added value considering the specified user goals and tasks". For example, the *Shopping Cart Basic Information* (2.1.1) subconcept combines two attributes whose *names* (recall Figure 3) are *Line item information completeness* (2.1.1.1), and *Product description appropriateness* (2.1.1.2), respectively (in Figure 4 they are also highlighted). In turn, the definition to the 2.1.1.1 attribute is "the extent to which the line item information coverage is the sufficient amount of data to an intended user goal", i.e., final users expect to have suitable information as title, author, price, quantity, added on date, and availability in order to accomplish the tasks effectively.

Table 1. External quality requirements (with regard to usability and content) for a shopping cart. EI = Elementary Indicator; P/GI = Partial/Global Indicator

External Quality Requirements	**EI value**	**P/GI value**
Global Quality Indicator		**61.97%**
1 Usability		**60.88%**
1.1 Understandability		83%
1.1.1 Shopping cart icon/label ease to be recognized	*100%*	
1.1.2 Information grouping cohesiveness	*66%*	
1.2 Learnability		51.97%
1.2.1 *Shopping cart help*	*50%*	
1.2.2 *Predictive information for link/icon*	*66%*	
1.2.3 Informative Feedback		41.5%
1.2.3.1 Continue-buying feedback	*66%*	
1.2.3.2 Recently viewed items feedback	*0%*	
1.2.3.3 Proceed-to-check-out feedback	*100%*	
1.2.3.4 User current status feedback	*0%*	
1.3 Operability		49.50%
1.3.1 Shopping cart control permanence	*100%*	
1.3.2 Expected behaviour of the shopping cart controls (just for the delete control in this study)	*50%*	
1.3.3 Accessibility of Controls		
1.3.3.1 *Support for text-only version of controls*	*0%*	
2 Content		**63.05%**
2.1 Information Suitability		63.05%
2.1.1 Shopping Cart Basic Information		50%
2.1.1.1 Line item information completeness	*50%*	
2.1.1.2 Product description appropriateness	*50%*	
2.1.2 Other Contextual Information		76.89%
2.1.2.1 Shipping and handling costs information completeness	*100%*	
2.1.2.2 Applicable taxes information completeness	*100%*	
2.1.2.3 Return policy information completeness	*33%*	

5.2.2 Design and Execution of the Measurement

First, for each attribute (leaf) of the requirement tree we have to design (or select) one metric, accordingly. For example, for the *Line item information completeness* attribute we have designed, taking into account the Amazon shopping cart as reference and previous tests we made, a direct metric named *Degree of completeness to the line item information* whose scale specifies three categories considering an ordinal scale type:

0. *Incomplete,* it has less information than category 1, or no information at all.
1. *Partially complete*, it only has title, price, quantity, and sometimes availability fields;
2. *Totally complete*, it has title, author, price, quantity, added on date, and availability.

The specification of the measurement method is objective and the data collection can be made observationally or maybe automated by a tool.

Second, once all the metrics were designed, we should perform the implementation of the measurement as discussed in Subsection 3.1. The measure value for the above metric is 1 w.r.t. the Cuspide's shopping cart instance. It is worth mentioning that we not only record the measured value

but also metadata like metric version, scale, scale type, specification and type of the measurement method, software tool (if any), accuracy, measurement time stamp, among others. In this way, metrics and measures can be repeatable, comparable and meaningful; this is, meaningful and purpose-oriented for a specific information need.

As commented in Subsection 3.2, the value of a particular metric will not represent the elementary requirement's satisfaction level, so we have to design an elementary indicator, which produce after enactment an elementary indicator value.

5.2.3 Design and Execution of the Evaluation

The following WebQEM step is the design of one elementary indicator per attribute of the requirement tree. For the 2.1.1.1 attribute, the elementary indicator named *Performance Level of the line item information completeness* will interpret the metric's value of it. Note that an elementary indicator interprets the level of satisfaction of this elementary nonfunctional requirement. Therefore, a new scale transformation and decision criteria (in terms of acceptability ranges) should be defined. In our study, we used three agreed acceptability ranges in a percentage scale: a value within 40-70 (a marginal –yellow- range) indicates a need for improvement actions; a value within 0-40 (an unsatisfactory –red- range) means changes must take place with high priority; a score within 70-100 indicates a satisfactory level –green- for the analyzed attribute.

Table 1 shows an elementary indicator value of 50% for the 2.1.1.1 attribute of the Cuspide shopping cart, taking into account that a measure value of 1 mapped to 50% and a value of 2 mapped to 100 percent of satisfaction (this mapping represents the *elementary model* depicted in Figure 3). Once all the elementary indicators were designed and calculated considering the same acceptability levels –but not necessarily the same elementary models-, the next step is the design and execution of the global evaluation.

Ultimately, the global evaluation problem can be defined as a problem of determining the extent to which a given entity fulfills a set of requirements for a given information need. The final outcome of the evaluation is a global indicator conveying the overall fulfillment of all requirements. Therefore, in order to enact a concept model for elementary, partial and global indicators, an *aggregation model* and decision criteria must be selected. The quantitative aggregation and scoring models aim at making the evaluation process well structured, objective, and comprehensible for evaluators. For example, if our function is based on a *linear additive scoring model*, the aggregation and computation of partial and global indicators (P/GI), considering relatives *weights* (W) is based on the following *specification*:

$$P/GI = (W_1\ EI_1 + W_2\ EI_2 + ... + W_m\ EI_m) \quad (1)$$

such that if the elementary indicator (EI) is in the percentage scale and unit, the following holds:

$$0 \leq EI_i \leq 100$$

and the sum of weights for an aggregation block must fulfill $(W_1 + W_2 + ... + W_m) = 1$ if $W_i > 0$; for i = 1 ... m, where m is the number of subconcepts at the same level in the tree's aggregation.

The basic arithmetic aggregation operator for the inputs in Eq. 1 is the plus (+) connector. Other non-linear aggregation models can be used in WebQEM such as logic scoring of preference, fuzzy model, and neural models, among others [19]. Particularly, in the present case study, we have selected the scoring model specified by Eq. 1. By applying equal weights –which means equal relative importance– and the + operator, we have related the hierarchically grouped attributes, sub-concepts,

and concepts of the instantiated external quality model accordingly, yielding in the end the partial and global indicator values shown in the rightmost column of Table 1.

5.2.4 Analysis and WMR Recommendation

We have already made one evaluation in order to understand the current state of the Cuspide shopping cart. At this point, we should analyze the results, draw conclusions and recommend improvements. We can see that many indicators are below the threshold of the satisfactory –green– acceptance range, so many attributes of usability and content of the Cuspide shopping cart may benefit from improvement.

The output of the analysis and recommendation activity is a list of attributes with unsuitable (yellow or red) indicator values, and for each one, a suggestion for improvement. In the integrated approach that combines WebQEM with WMR, when the improvement does not require a change in the application's behavior, a list of one or more WMRs is suggested.

For instance, Cuspide should plan changes in the *Shopping Cart Basic Information* sub-characteristic (ranked 50%) mainly in the *Line item information completeness* (2.1.1.1), and *Product description appropriateness* (2.1.1.2) attributes. For example, the *Line item information completeness* should have at least the author's name besides the title of the item, since it is not possible to distinguish between two or more items with the same starting title (e.g., as shown in Figure 4 with the title "Ingenieria de Software").

These discussed attributes, as currently designed, are not totally suitable because the user cannot operate effectively (for lack of information in the shopping cart), or is required to do more clicks than necessary to go to the detailed product information. On the other hand, the delete control is not operable as the expected behavior for this kind of widget diminishing its usability.

In order to select the appropriate WMR, we have defined a correspondence between WMRs and quality attributes that each one may improve. Table 2 shows the result of this correspondence, with WMRs to the left of the table and the attributes they improve to the right. The correspondence shown in this table should be agreed by experts at least once; then a purposely designed catalogue can be able to record this experience for ulterior reuse in a selection process.

Table 2. Mapping between WMRs and attributes (shown in Table 1) that can be applied to improve the external quality. NM = Navigation Model; PM = Presentation Model

WMRs that may apply	**Attributes that may improve**
Add Node Attribute (NM)	1.1.1 / 1.2.3.3 / 1.2.3.4 / 2.1.1.1 / 2.1.1.2 / 2.1.2.1 / 2.1.2.2 / 2.1.2.3
Add Category (NM)	1.1.2 / 1.2.1
Add Node Operation (NM)	1.2.3.3 / 1.3.1
Add Index (NM)	1.2.1
Add Guided Tour (NM)	1.2.1
Anticipate Target (NM)	1.2.2
Enrich Index (NM)	1.2.3.1 / 2.1.1.2
Introduce History (NM)	1.2.3.2
Multiply Category (NM)	1.1.2 / 1.2.1
Recategorize Item (NM)	1.1.2 / 1.2.1
Turn Info into Link (NM)	1.2.3.2 / 2.1.1.2 / 2.1.2.2
Add Widget (PM)	1.1.1
Replace by Text (PM)	1.3.3.1
Replace Widget (PM)	1.3.2

5.2.5 Applying WMR

After the analysis and recommendation step, we are ready to start refactoring the navigation and presentation models. In our case study, we may apply at least three refactorings that we described as examples in Section 4.3. First, in order to improve the *Line item information completeness (2.1.1.1)*, we have applied the refactoring ADD NODE ATTRIBUTE to incorporate in the shopping cart node the author's name for each item in the list. Second, note that it is not possible to navigate to the pages showing the detailed information of the items in the shopping cart. In this case, we can improve the *Product description appropriateness (2.1.1.2)* by applying the refactoring TURN INFORMATION INTO LINK. Third, to correct the problem with the *Expected behaviour of the shopping cart controls (1.3.2)* attribute, specifically the unexpected behavior of the delete item control, we have applied the refactoring REPLACE WIDGET.

5.2.6 Assessing Quality Improvements

The last activity of the integrated improvement process is to assess the benefits of applying WMR, by repeating the measurement and evaluation. Nevertheless, there is no need to measure and evaluate every single quality attribute again, but only those that were affected by the applied WMRs. The mapping presented in Table 2 helps us one more time to find these attributes, which in our case study are: the *Line item information completeness (2.1.1.1)*, the *Product description appropriateness (2.1.1.2)* and the *Expected behaviour of the shopping cart controls (1.3.2)*.

Figure 5 shows the shopping cart resulting from applying the three WMRs discussed above: ADD NODE ATTRIBUTE, TURN INFORMATION INTO LINK, and REPLACE WIDGET. As a result, we can predict the total satisfaction (100%) of the affected attributes and no further evaluation and refactoring phases are needed.

Figure 5: The Cuspide shopping cart after applying some refactorings.

5.3 Value of the Integrated Approach

As stated before, our research aims at: (a) recommending WMRs that may improve the external quality or quality in use of a Web application; and (b) integrating quality evaluation in the refactoring process.

There are two ways in which we can face the refactoring activity during the development cycle: as an informal improvement process or in the context of a structured evaluation framework.

In the first case, we analyze our application (either by collecting users' feedback or by carefully inspecting its functionality) and find opportunities for refactoring. In fact, this is the way in which refactoring has been applied so far in the software community. When the designer is aware of a good catalogue of possible refactorings, the process is simplified. As we have shown in Section 5.3, even simple refactorings may produce an important gain in the application's quality.

The second case (using a structured evaluation framework) arises when we are able to formally perform an evaluation before and after the refactoring. Moreover, by performing the evaluation of the entity to be refactored before and after the process, we can quantify and justify the quality gain, independently of the chosen lifecycle. Therefore, the incremental quality improvement can be evaluated and/or predicted.

The most important aspect of this strategy is that, ultimately, we can map atomic or composite refactorings to quality attributes. In other words, associated with each meaningful attribute, there are one or more refactorings that may be applied to meet this elementary non-functional requirement. Moreover, at the organization level, we can have a catalogue of atomic and composite WMRs (as commented in Section 4) and eventually a mapping to a catalogue of attributes. By knowing beforehand the impact of the transformations, we could estimate the improvement gain.

As previously mentioned we were able to make a correspondence between WMRs and attributes that influence the external quality of our case study. Of course we might not have to apply all the WMRs to all the Cuspide shopping cart attributes listed in Table 2. Some of them may not need real improvement, e.g., those in which the actual elementary indicator value is 100%. However, for those attributes of the shopping cart which are weak or absent (as it was shown by performing the evaluation in our study), we can predict, after applying a focused cost-effective WMR, a total level of satisfaction of all these requirements.

To summarize, we can say that our integrated approach provides two main contributions:

- adding value to WMR from the use of WebQEM, since this measurement and evaluation method allows detecting sources of problems with the quality of a Web application, thus recommending WMRs from a more formal and systematic approach than just following "bad smells"; furthermore, applying WebQEM after refactoring allows to actually make a real assessment of the improvements;
- adding value to WebQEM from the use of WMR, since applying small and focalized changes allows keeping track of the specific quality attributes improved, thus reducing the amount of measurements needed afterwards, in the same way that conventional refactoring narrows down the amount of testing needed afterwards.

6 Related Work

Our research differs from existing works in the refactoring field in three aspects: the subject, the intent, and the underlying strategy. Regarding the subject we deal with the navigational and interface models of Web design methods, while existing literature works either at the code level or at the implementation design level (e.g., by refactoring on UML diagrams). Even for those Web design methods whose notations are based on UML-like diagrams [5, 16], our refactorings are different from those in [34]. Regarding the intent, Web model refactorings are aimed at improving the user experience and not to ease the maintenance or quality of code. Finally, regarding the underlying strategy, our approach differs from others in the sense that it is a purpose-oriented Web model refactoring driven by non-functional information needs both at organization and design level. In this direction, we strongly integrate a measurement and evaluation methodology (WebQEM) –based on an ontological framework for such end, with a Web model refactoring approach.

Ricca and Tonella [27] have worked on code restructuring for Web applications and use some metrics for measurement. They define different categories of restructuring, like syntax update, internal page improvement, and dynamic page construction. Refactoring differs from restructuring in that the latter implies larger transformations that are usually run in batch mode by applying certain rules. Instead, refactorings are smaller and applied interactively. However, one main difference with our work is that their transformations apply on the source code, in this case, HTML, PHP and/or Javascript, while no definition is given of what is the behavior that the proposed transformations preserve. Another difference is that we classify atomic or composite refactoring with regard to the impact on usability, content, etc. Specifically, we can observe the set of navigation and presentation model refactorings that may improve learnability, operability, attractiveness, information suitability, among other non-functional requirements, at the levels of characteristic and measurable attribute.

On the other hand, Ping and Kontogiannis apply refactoring at the level of hypermedia links, i.e., to the navigational structure of the application [26]. They propose an algorithm to cluster links into several types and group Web pages according to these link types. Applying this technique should provide a roadmap for the identification of controller components of a controller-centric architecture. Although their target is the navigational structure of a Web application, they do not provide the mechanics to apply the transformation, but only a first step to recognizing where to apply them. In addition, in a recent work, Ivkovic et al. [11] include in their refactoring strategy a soft-goal hierarchy identification step. Even though this work represents a valuable contribution, a sound framework to justify measurement and evaluation results for analysis and recommendation of quality improvements is missing.

In the Web Engineering field, other authors have dealt with the problem of improving the quality of Web software. In [4] the authors propose a Model Driven approach for improving the navigability of Web applications. The approach shares with ours the use of a solid underlying quality model to measure the quality of the navigational model; it is different in that it uses a transformation approach (based on a software tool) to improve the application when necessary. In this sense this approach is more ambitious than ours, which so far is basically hand-based. Our ideas meanwhile cover not only the navigational but also the interface and interaction aspects of the applications and uses refactoring in the same spirit as well-known agile methodologies.

Lastly, we have developed a measurement and evaluation approach which is made up of three tenets, i.e. the process, the conceptual framework, and the methodology. The ontology underlying our

measurement and evaluation framework has similarities to the one presented in [8] and then slightly refined by [9]. However in [17] we have modeled some terms (e.g., elementary indicator, global indicator, etc.) and some relationships (e.g, measurement and measure, metric and indicator, among others) that differ semantically with those proposed in [9]. Our ontology was also to a great extent akin to the terminology used in WebQEM since late 90s.

7 Concluding Remarks

In this paper we have presented our proposal to continuously improve the external quality of Web applications during their entire life-cycle. The approach is based on the use of WMR combined with WebQEM, a mature method for assessing the quality characteristics of a Web application. We defined WMRs as those refactorings that can be applied to the navigation and presentation models of a Web application with the purpose of improving its external characteristics, while preserving its functionality and behavior.

We showed how to incorporate the WebQEM quality evaluation method in the process of refactoring in order to: (*i*) specify quality attributes, (*ii*) measure and evaluate them before refactoring to recommend what refactoring to apply, and (*iii*) evaluate them after refactoring to assess the improvements. We described the integrated process with a concrete case study showing how a typical shopping cart in an e-commerce site can improve its usability by applying a representative set of WMRs.

Our current work on the subject is being devoted to extending our catalogue of WMRs, and studying the dependence and relationships between atomic refactorings to produce more complex, composite refactorings. In the integration with WebQEM, extending the catalogue of WMRs also requires enriching the mapping between the new WMRs and quality attributes of a Web application, to improve the recommendation process. A further research objective we are pursuing is the development of tool support both for WMR and quality assessment, for the OOHDM [31] and UWA [32] methods, by integrating such features in their model-driven development tools.

References

1. Beck, K. (2002) *Test-Driven Development*. Addison-Wesley.
2. Beck, K. (2006) *Extreme Programming Explained*. Addison-Wesley.
3. Boger, M.; Sturm, T.; Fragemann, P. (2003) Refactoring Browser for UML. *LNCS 2591*, Springer 2003, pp. 366-377.
4. Cachero, C.; Melia, S.; Genero, M.; Poels, G.; Calero, C. (2007) Towards improving the navigability of Web applications: a model-driven approach. European Journal of Information Systems 16, pp. 420-447.
5. Ceri, S., Fraternali, P., Bongio, A. (2000). Web Modeling Language (WebML): a Modeling Language for Designing Web Sites. Proc WWW9 Conference, Amsterdam, NL.
6. Covella, G. Olsina, L. (2006). Assessing Quality in Use in a Consistent Way, *In Proc. of ACM, Int'l Congress on Web Engineering, (ICWE05)*, SF, USA, pp.1-8.
7. Fowler, M. (2000). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley.
8. García, F.; Ruiz, F.; Bertoa, M. F.; Calero, C.; Genero, M.; Olsina, L.; Martín, M.; Quer, C.; Tondori, N.; Abrahao, S.; Vallecillo, A. and Piattini, M, (2004) An ontology for software measurement, Technical Report UCLM DIAB-04-02-2, Computer Science Department, University of Castilla-La Mancha, Spain, (In Spanish).
9. Garcia, F.; Bertoa, M F.; Calero, C.; Vallecillo, A.; Ruiz F.; Piattini, M.; Genero, M. (2005) Towards a consistent terminology for software measurement. Information and Software Technology 48(8), pp. 631-644.

10. Garrido, A.; Rossi, G.; Distante, D. (2007). Model Refactoring in Web Applications. *In Proceedings of the 9th International Symposium on Web Site Evolution* (WSE 2007: Oct. 05-06, 2007; Paris, France). Los Alamitos, CA: IEEE Computer Society Press.
11. Ivkovic, I.; Kontogiannis, K. (2006). A Framework for Software Architecture Refactoring using Model Transformations and Semantic Annotations. *In Proceedings of the IEEE Conference on Software Maintenance and Reengineering (CSMR'06)*, pp. 135-144.
12. ISO/IEC 14598-1 (1999). "International Standard, Information technology - Software product evaluation - Part 1: General Overview".
13. ISO/IEC 9126-1 (2001). "International Standard, Software Engineering - Product Quality - Part 1: Quality Model".
14. ISO/IEC 15939 (2002). "Software Engineering - Software Measurement Process".
15. Kerievsky, J. (2005). *Refactoring to Patterns*. Addison-Wesley.
16. Koch, N., Kraus, A. (2002). The Expressive Power of UML-based Web Engineering. *In Proc. of 2nd Int'l Workshop on Web Oriented Software Technology (IWWOST´02)* at ECOOP´02. Málaga, Spain, pp. 105-119.
17. Martín, M.; Olsina, L., (2003) Towards an Ontology for Software Metrics and Indicators as the Foundation for a Cataloging Web System, *In proceed. of IEEE Computer Society (1st Latin American Web Congress),* Santiago de Chile, pp 103-113, ISBN 0-7695-2058-8.
18. Olsina, L; Rossi, G.; Garrido, A.; Distante, D.; Canfora, G. (2007) Incremental Quality Improvement in Web Applications Using Web Model Refactoring, *LNCS 4832, Springer 2007, pp. 411-422, 1st Int. Workshop on Web Usability and Accessibility (IWWUA´07)*, Nancy, France.
19. Olsina, L., Papa, F., Molina, H. (2007) How to Measure and Evaluate Web Applications in a Consistent Way. Chapter Thirteenth in Springer Book, Human-Computer Interaction Series, titled *Web Engineering: Modelling and Implementing Web Applications*; Rossi, Pastor, Schwabe, and Olsina (Eds.), pp. 385–420.
20. Olsina, L., Covella, G., Rossi, G. (2006).Web Quality. Chapter fourth in Springer Book titled *Web Engineering*, Emilia Mendes and Nile Mosley (Eds), pp. 109–142.
21. Olsina, L., Martín, M. (2004). Ontology for Software Metrics and Indicators, *Journal of Web Engineering*, Rinton Press, US, Vol 2 Nº 4, pp. 262-281, ISSN 1540-9589
22. Olsina, L., Rossi G., (2002). Measuring Web Application Quality with WebQEM, *IEEE Multimedia*, 9(4), pp. 20-29.
23. Olsina, L., Lafuente G, Rossi G (2000) E-commerce Site Evaluation: a Case Study, *LNCS 1875 of Springer, 1st Int'l Conference on Electronic Commerce and Web Technologies, EC-Web 2000*, London, UK, pp. 239-252.
24. Olsina, L., Godoy, D., Lafuente, G., Rossi, G. (1999). Assessing the Quality of Academic Websites: a Case Study, *New Review of Hypermedia and Multimedia (NRHM) Journal*, Taylor Graham Publishers, UK/USA Vol. 5, pp. 81-103
25. Opdyke, W. (1992). Refactoring Object-Oriented Frameworks. Ph.D.Thesis, University of Illinois at Urbana-Champaign.
26. Ping, Y. and Kontogiannis, K. (2004). Refactoring Web sites to the Controller-Centric Architecture. *In Proc. of the European Conf. on Software Maintenance and Reengineering (CSMR'04)*. Tampere, Finland.
27. Ricca, F. and Tonella, P. (2005). Program Transformations for Web Application Restructuring. *In Web Engineering: Principles and Techniques*. Woojong Suh (Eds.). Ch. XI: pp. 242-260.
28. Ricca, F., Tonella, P., Baxter, Ira D. (2001). Restructuring Web Applications via Transformation Rules. *SCAM*, pp. 152-162.
29. Roberts, D. (1999). Eliminating Analysis in Refactoring. Ph.D. Thesis, University of Illinois at Urbana-Champaign.
30. Rossi, G., Urbieta, M., Ginzburg, J., Distante, D., Garrido, A. (2008) Refactoring to Rich Internet Applications. A Model-Driven Approach. In Proc. of 8th International Conference on Web Engineering, (ICWE 2008), NY, USA.
31. Schwabe, D., Rossi, G. (1998). An Object Oriented Approach to Web-Based Application Design. *Theory and Practice of Object Systems* 4(4), Wiley and Sons, NY.
32. UWA Consortium, (2002). Ubiquitous Web Applications. *Proceedings of the eBusiness and eWork Conference e2002*; Prague, Czech Republic.
33. Van Duyne, D., Landay, J., Hong, J. (2003). *The Design of Sites*. Addison-Wesley.
34. Van Gorp, P., Stenten, H., Mens, T., Demeyer S. (2003). Towards automating source-consistent UML Refactorings. *In Proceedings of the 6th Int'l Conference on UML.*
35. Zhang, J., Lin, Y., Gray, J. (2005). Generic and Domain-Specific Model Refactoring using a Model Transformation Engine. Chapter 9, In Springer Book *Model-driven Software Development*, S. Beydeda, M. Book, and V. Gruhn, (Eds.), pp. 199-218.

Appendix

Table 3. Measurement and Evaluation Ontology: Abridged Glossary of Terms (see [19] for more details).

Main Concepts	Description
NF Requirement Terms	A measurable physical or abstract property of an entity category.
Attribute (syno Property, Feature)	
Information Need	Insight necessary to manage objectives, goals, risks, and problems.
Calculable Concept (syno Measurable Concept in [14])	Abstract relationship between attributes of entities and information needs.
Concept Model (syno Factor , Feature Model)	The set of sub-concepts and the relationships between them, which provide the basis for specifying the concept requirement and its further evaluation or estimation.
Entity (syno Object)	A concrete object that belongs to an entity category.
Entity Category	Object category that is to be characterized by measuring its attributes.
Requirement Tree	Note. A requirement tree is a constraint to the kind of relationships among the elements of the concept model, regarding the graph theory
Measurement Terms	A metric of an attribute that does not depend upon a metric of any other attribute.
Direct Metric (syno Base, Single Metric)	
Indirect Metric (syno Derived, Hybrid Metric)	A metric of an attribute that is derived from metrics of one or more other attributes.
Measure	The number or category assigned to an attribute of an entity by making a measurement.
Measurement	An activity that uses a metric definition in order to produce a measure's value.
Measurement Method (syno Counting Rule, Protocol)	The particular logical sequence of operations and possible heuristics specified for allowing the realisation of a direct metric description by a measurement.
Metric	The defined measurement or calculation method and the measurement scale.
Scale	A set of values with defined properties. Note. The scale type depends on the nature of the relationship between values of the scale. The scale types mostly used in Web engineering are classified into nominal, ordinal, interval, ratio, and absolute.
Unit	A particular quantity defined and adopted by convention, with which other quantities of the same kind are compared in order to express their magnitude relative to that quantity.
Evaluation Terms	Thresholds, targets, or patterns used to determine the need for action or further investigation, or to describe the level of confidence in a given result.
Decision Criteria (syno Acceptability Levels)	
Elementary Indicator (syno Elementary Preference, Criterion)	An indicator that does not depend upon other indicators to evaluate or estimate a calculable concept.
Elementary Model (syno Elementary Criterion Function)	Algorithm or function with associated decision criteria that model an elementary indicator.
Evaluation (syno Calculation)	Activity that uses an indicator definition in order to produce an indicator's value.
Global Indicator (syno Global Preference, Criterion)	An indicator that is derived from other indicators to evaluate or estimate a calculable concept.
Global Model (syno Scoring, Aggregation Model or Function)	Algorithm or function with associated decision criteria that model a global indicator.
Indicator (syno Criterion)	The defined calculation method and scale in addition to the model and decision criteria in order to provide an estimate or evaluation of a calculable concept with respect to defined information needs.
Indicator Value (syno Preference Value)	The number or category assigned to a calculable concept by making an evaluation.