Incremental Quality Improvement in Web Applications Using Web Model Refactoring

Luis Olsina[1], Gustavo Rossi[2], Alejandra Garrido[2],
Damiano Distante[3], and Gerardo Canfora[3]

[1] GIDIS_Web, Universidad Nacional de La Pampa, Argentina
[2] LIFIA, Universidad Nacional de La Plata and CONICET, Argentina
[3] RCOST, University of Sannio, Italy
olsinal@ing.unlpam.edu.ar,
{gustavo,garrido}@lifia.info.unlp.edu.ar,
{distante,canfora}@unisannio.it

Abstract. Web applications must be usable and accessible; besides, they evolve at a fast pace and it is difficult to sustain a high degree of external quality. Agile methods and continuous refactoring are well-suited for the rapid development of Web applications since they particularly support continuous evolution. However, the purpose of traditional refactorings is to improve internal quality, like maintainability of design and code, rather than usability of the application. We have defined Web model refactorings as transformations on the navigation and presentation models of a Web application. In this paper, we demonstrate how Web model refactorings can improve the usability of a Web application by using a mature quality evaluation approach (WebQEM) to assess the impact of refactoring on some defined attributes of a Web product entity. We present a case study showing how a shopping cart in an e-commerce site can improve its usability by applying Web model refactorings.

Keywords: refactoring, Web applications, usability, quality evaluation.

1 Introduction

The evolution of Web applications (WAs) is driven by a myriad of different factors: new requirements (stable and volatile), users' feedback, new technologies giving the chance to change the look and feel or the interaction style of the application, etc. In all cases, this evolution usually follows unpredictable patterns that imply a constant pressure on development teams. Agile methods have emerged to help developers cope with, and even welcome, continuous change in requirements [1]; as such, these methods are particularly suitable for developing WAs. Refactoring is one of the fundamental practices of agile development used to add flexibility and extensibility before introducing new functionality [3].

Refactoring was defined in the context of object-oriented systems to "factor out" new abstractions and perform other small transformations to the source code of an application without changing its behavior [11]. These transformations aim at improving the design of the code, making it more reusable and flexible to subsequent

M. Weske, M.-S. Hacid, C. Godart (Eds.): WISE 2007 Workshops, LNCS 4832, pp. 411–422, 2007.

semantic changes. Though refactorings are performed in small steps, they are usually composable, yielding larger transformations that improve readability, reusability and maintainability of a system [15]. Refactoring to patterns has also been proposed to help keep the balance between under-engineering (continuously adding new functionality without a previous clean-up) and over-engineering (applying design patterns to create overly complex designs) [6].

In the context of WAs, refactoring may not only be applied to improve internal quality, but also to enhance navigability and presentation, which are external qualities influencing usability. In [4] we have defined the concept of Web model refactoring (WMR), i.e., refactoring applied to the navigation and presentation models of a Web application. WMRs aim at improving the application's usability by small transformations in the application's navigational topology, and/or interface look and feel. Additionally, WMRs guide the introduction of Web patterns [18] into the application's structure. In order to assess how WMRs improves usability, we use a well-known Web quality evaluation approach, namely WebQEM [8], and test the application's quality features before and after refactoring.

As an example that we will elaborate in this paper, Fig. 1.a shows a reduced version of the Amazon shopping cart and Fig. 1.b shows the same cart with some added information and operations. In our research we want to identify this kind of transformations and be able to measure the associated quality improvement, if any.

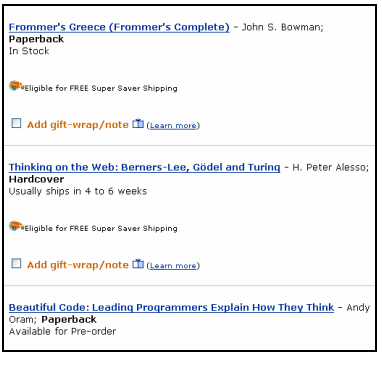

Fig. 1.a. Basic shopping cart

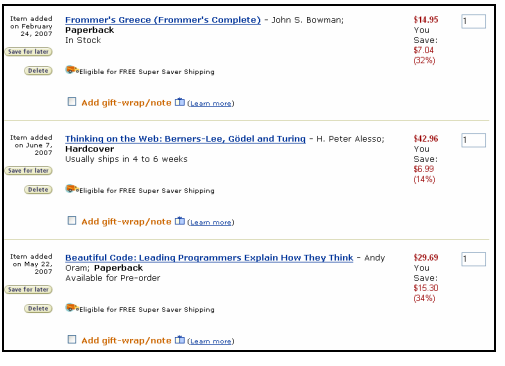

Fig. 1.b. Enhanced shopping cart

The main contributions of this paper are the following: (a) we propose WMR as a way to incrementally improve the external quality of a WA from the final user viewpoint; (b) we show how to incorporate quality assessment in the process of refactoring; and, (c) we demonstrate how WMR improves usability on a particular case study.

The structure of the paper is as follows. Section 2 summarizes the concept of WMR and the WebQEM quality evaluation approach. Section 3 shows how to apply WebQEM in the context of WMR by using a simple but meaningful case study. Section 4 presents related work and Section 5 concludes the paper and describes some further work.

2 Background

2.1 Web Model Refactoring

Refactoring is a technique that assists developers in the process of continuous improvement of source code or design models of an application [20]. We are interested in defining refactorings for the design models of a WA. Well-known design methods agree on the definition of a three-stage design process for WAs, resulting in the definition of three models: *application model*, *navigation model* and *presentation model* [2, 7, 16, 17]. While refactorings applied to the application model are similar to those already in the literature [19, 20], we have defined WMRs as those refactorings that can be applied to the navigation and presentation models of a WA [4].

2.1.1 Intent, Scope and Granularity

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 maintainability [11]. Web code refactorings, e.g., those applied to the source code or HTML structures, are outside the scope of this paper but discussed elsewhere [13, 14]. Similarly, WMRs apply model transformations that affect the way in which the application presents contents, enables navigation through contents and provides interaction capabilities [4], but do not change the semantics as defined for these models.

Since WMRs transform entities that are perceived by the user, their changes are directly reflected in the way the final user may interact with the WA. Therefore, the intent of WMRs is to enhance usability [9]. These refactorings 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 [6], WMRs might be also driven by Web patterns [18] and therefore produce the same well-known benefits of the patterns they introduce.

Navigation refactorings aim at improving the application's navigability by small transformations of its navigation model. This model is usually described with a navigational diagram composed of nodes, links, indexes and other access structures. Behavior defined by the navigation model of a WA 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. Navigation model refactorings may thus change, among others: the contents available in a node, the set of outgoing links of a node and the user operations accessible from a node. Alongside, navigation model refactorings have to preserve the set of possible operations and their semantics, and the navigability of the set of nodes [4]. Preserving the navigability of the set of nodes means that existing nodes may not become unreachable though the set may be augmented. Moreover, this type of refactorings should not introduce data, relationships or operations that are not in the application model.

The presentation 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 the result of user interaction. Presentation model refactorings may thus transform the look and feel of a page by changing the

position of widgets or the interface effects, by adding new widgets or replacing them to enhance understanding, etc. At same time, the behavior defined by the presentation model, which must be preserved by the refactorings on this model, concerns with the actions that the user may trigger in a page, including both operations and links activation of the underlying nodes.

Most navigation model refactorings may imply other refactorings 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.

2.1.2 Examples

To illustrate the ideas presented above, we next describe some representative WMRs using a simplified template comprising: type of refactoring (navigation or presentation), motivation, mechanics and example. Other refactorings, including composite refactorings, can be found in [4].

ADD INFORMATION (NAVIGATION)

Motivation: we may eventually find the need to display more information than what is currently on a page. The information may come from different sources and have different purposes: it may be data extracted from the application model or obtained from the navigation model itself; it may be information added with the purpose of attracting customers or to help during navigation. This refactoring may be used to introduce patterns like *Clean Product Details* [18] to add details about products in an e-commerce site. With the aim of attracting customers, we may introduce *Personalized Recommendations* [18] or rating information.

Mechanics: the mechanics varies according to the different sub-intents above. In the most general case: add an attribute to a node class in the navigation model where the information is to be added. 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 behind the page appearing in Fig. 1.a, to show 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 Fig. 1.b. It is worth noting that the information added to the shopping cart node is already available from the application model.

TURN INFORMATION INTO LINK (NAVIGATION)

***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. A common example occurs when checking the status of the shopping cart during the process of buying some products in an e-commerce website. 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 to improve it.

Mechanics: select the portion of the information about the target item that better distinguishes it. In the navigational diagram, add a link from the node representing the

intermediate results to the nodes showing detailed information on the items to review; 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.

REPLACE WIDGET (PRESENTATION)
Motivation: the presentation model describes, for each node in the navigation model, the kind of widgets that display each data attribute and the widgets 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 to enable users select one or more items from a list to 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". Thus, 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 Fig. 3, checking any box under the title "Borrar" automatically deletes the item from the cart, which is 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 Fig. 4).

2.2 The Web Quality Evaluation Method (WebQEM) Approach

WebQEM [8] is an evaluation method for WAs, i.e., a method for the inspection of characteristics, sub-characteristics (named calculable concepts and sub-concepts in Fig. 2), and attributes stemming from a quality model for WAs. WebQEM relies on a set of well-defined metrics and indicators for measurement and evaluation, in order to give recommendations for improvement. The main parts of the measurement and evaluation framework (named INCAMI [10], which stands for *Information Need*, *Concept model*, *Attribute*, *Metric* and *Indicator*), and the WebQEM method that instantiates it are, namely:

- The *non-functional requirements specification component,* which deals with the definition of the *Information Need* and the specification of requirements by means of one or more *Concept Models* -see Fig. 2. (Note that a concept model can be instantiated in external quality, quality in use models, among many others).
- The *measurement design and execution component,* which deals with the specification of concrete *Entities* to be measured, the metrics selection to quantify the attributes of the quality model, and the recording of the gathered measures; this component is centered in the *Metric* concept [10].
- The *evaluation design and execution component,* which deals with the definition of indicators, both elementary and global ones, decision criteria and aggregation models that will help to enact and interpret the selected concept model; this component is centered in the *Indicator* concept (see [10] for more details).

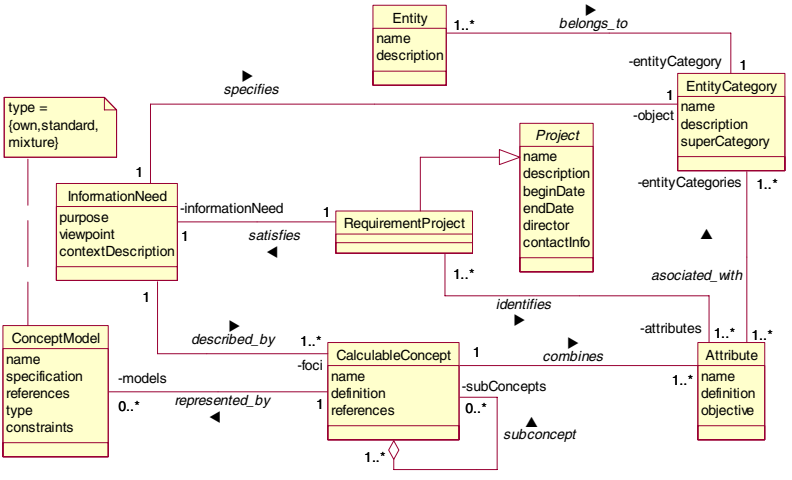

Fig. 2. Key terms and relationships that intervene in the non-functional requirements definition and specification component, which is instantiated by WebQEM

3 A Strategy for Incremental External Quality Improvement

Our proposal to continuously improve the external quality of a WA, such as its usability, is to enrich the lifecycle of a WA with a new activity: WMR. After each requirements/design/implementation cycle and before the next cycle begins, the design models of a WA should be inspected in order to find opportunities for refactoring. Thus, similarly to the well-known advantages of traditional refactoring [3, 20], the quality of the design will be incrementally improved, as well as the external quality. We also propose an additional activity to assess the benefits of applying WMR by evaluating the application's quality before and after refactoring. This is a first step to systematize the process of WMR as a quality-driven activity.

3.1 The Shopping Cart Case Study

The external quality (e.g., the usability and content) of the Cuspide shopping cart can be improved by systematically applying WMR. Instead of using a set of "good practices" for shopping carts, we used the Amazon.com shopping cart as a reference for desirable requirements and quality attributes of a shopping cart. In fact, Amazon is certainly a well-known instantiation of good practices for this type of application component. Referring back to Fig. 2, we can state that given an *entity* (the Cuspide shopping cart), the *information need* can be specified by its *purpose* (evaluate and compare), from the *user viewpoint* (developer), 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 *sub-concepts* (usability and content), which can be *represented by* a *concept model* (external quality model) and associated *attributes* (as shown in the left side of Table 1).

3.1.1 The External Quality Specification

To evaluate the impact of WMR, we have specified the external quality of the Cuspide shopping cart with regard to some usability and content attributes, contrasting it with some features of the Amazon shopping cart. In this sense, we carried out evaluation activities before and after applying WMRs. The external quality requirements that were assessed appear in the left column of Table 1. In addition, elementary, partial and global indicator values are shown for the Cuspide shopping cart before refactoring (as we analyse in the next sub-section). Many of these quality requirements were illustrated in [9], as well as the justification for the inclusion of the *Content* characteristic for assessing the quality of information in the Web.

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

External Quality Requirements	EI	P/GI
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 behavior of shopping cart controls	*50%*	
1.3.3 Controls Accessibility		
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 costs information completeness	*100%*	
2.1.2.2 Applicable taxes information completeness	*100%*	
2.1.2.3 Return policy information completeness	*33%*	

3.1.2 Design and Execution of the Measurement and Evaluation

Following the WebQEM's steps (outlined in Section 2.2), evaluators should design, for each attribute of the instantiated quality model, the basis for the measurement and evaluation process. This step is accomplished by defining each specific metric and

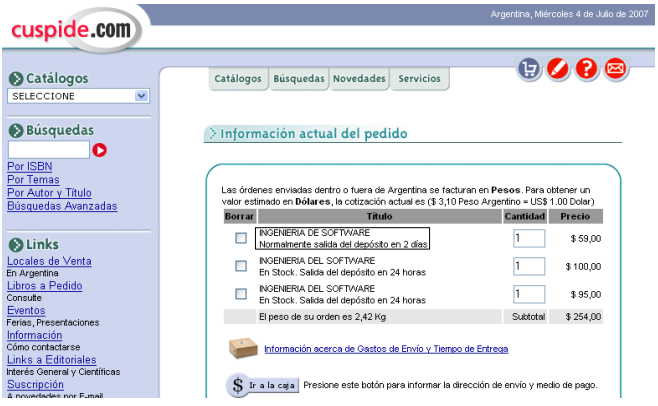

Fig. 3. Shopping cart in Cuspide (www.cuspide.com.ar) before refactoring

indicator for the given information need. Lastly, in the execution phase, we record the final values for each metric and indicator. The right columns of Table 1 show the indicators values, and Fig. 3 the entity to be measured.

For example, for the "*Line item information completeness*" attribute (coded 2.1.1.1 in Table 1), having the Amazon shopping cart as reference, we designed a direct metric named "*Degree of completeness of the Line item information*". It specifies three categories considering an ordinal scale type, namely: 0. *Incomplete; 1. Partially complete*, (i.e., it only has title, price, quantity, and sometimes availability fields); and 2. *Totally complete* (it has 1 plus author, added on date, and availability).

Moreover, an elementary indicator can be defined for each attribute of the requirement tree. For instance, for the previous attribute, the elementary indicator "*Performance Level of the Line item information completenes"* interprets the metrics value of the attribute. Note that an elementary indicator interprets the level of satisfaction of this elementary requirement. After this, a new scale transformation and decision criteria (in terms of acceptability ranges) are defined. In our study, we use three agreed acceptability ranges in a percentage scale: a value within 40-70 (a marginal range) indicates a need for improvement actions; a value within 0-40 (an unsatisfactory range) means changes must take place with high priority; a score within 70-100 indicates a satisfactory level for the analyzed attribute. Table 1 shows a value of 50% for the 2.1.1.1 attribute of the Cuspide shopping cart, taking into account that a value of 1 mapped to 50% and a value of 2 mapped to 100% of satisfaction.

Furthermore, to design and execute the global evaluation, we should select and apply an aggregation and scoring model [8]. In this case study, we used an additive scoring model, so applying weights and the sum operator we related the hierarchically grouped attributes, sub-concepts, and concepts accordingly, yielding in the end the partial and global indicators (the rightmost column of Table 1). Thus, decision-makers can analyze the results and give recommendations. We can see that many indicators are below the threshold of the satisfactory acceptance range; thus, many attributes of the external quality of the Cuspide shopping cart may benefit from improvement.

3.1.3 Applying WMR to the Example

As discussed above, Cuspide should plan changes in the *Shopping Cart Basic Information* sub-characteristic (ranked 50%) mainly in attributes 2.1.1.1 and 2.1.1.2. For example, the 2.1.1.1 attribute should have at least the author's name besides the title of the item, since, as shown in Fig. 3, it is not possible to distinguish between two or more items with the same starting title (e.g., "INGENIERIA DE SOFTWARE"). We may apply the refactoring ADD INFORMATION (see Section 2.1.2) to the shopping cart node to incorporate the author's name to each item in the list. Moreover, note that it is not possible to navigate to the pages showing the detailed information on the items in the shopping cart for further information. We can apply the refactoring TURN INFORMATION INTO LINK to solve this problem. The outcome of applying these two refactorings is shown in Fig. 4. As a result we can predict the total satisfaction (100%) of both attributes mentioned above. Fig. 4 also shows the result of applying the refactoring REPLACE WIDGET (see Section 2.1.2) to correct the unexpected behavior of the delete item control.

Fig. 4. The Cuspide shopping cart after refactoring

3.2 Discussion

As stated before, our research aims at: (a) presenting WMRs that may improve the external quality of a WA; and (b) integrating quality assessment 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. 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 attributes. In other words, associated with each meaningful attribute, there are one or more refactorings that may be applied to meet this requirement. Moreover, at the organization level, we can have a catalogue of WMRs and eventually a mapping to a catalogue of attributes. By knowing beforehand the impact of the transformations, we could estimate the improvement gain. In our case study, we were able to make a correspondence between refactorings and attributes that influence the external quality. Table 2 shows the result of this correspondence, with refactorings to the left of the table and the attributes they improve to the right.

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

Refactorings that may apply	Attributes that may improve
Add Information (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 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)	2.1.2.2
Add Widget (PM)	1.1.1
Replace by Text (PM)	1.3.3.1
Replace Widget (PM)	1.3.2

Of course we might not have to apply all the refactorings 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 which are weak or absent we can predict, after applying a focused cost-effective refactoring, a total level of satisfaction of all these requirements. Ultimately, our strategy for incremental improvement can be used both to predict the quality and to actually make the real assessment after refactoring.

4 Related Work

Our research differs from existing work in the refactoring field in three aspects: the subject, the intent, and the underlying strategy. Regarding the subject, we deal with the navigation and interface models of a WA, 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 [2, 7], our refactorings are different from conventional model

refactorings [19]. Regarding the intent, WMRs aim at improving the user's experience with the WA and not internal attributes such as maintainability. Finally, regarding the underlying strategy, our approach differs from others in that it integrates an evaluation methodology (WebQEM) in order to improve non-functional information needs.

Ricca and Tonella [13] have worked on code restructuring for WAs. 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. Another difference is that WMRs are defined to improve operability, attractiveness, information suitability, among other non-functional characteristics, at the levels of characteristic and measurable attributes.

On the other hand, Ping and Kontogiannis apply refactoring at the level of hypermedia links, i.e., to the navigational structure of the application [12]. 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 WA, 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. [5] 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.

5 Concluding Remarks and Further Work

In this paper we have presented our proposal to continuously improve the external quality of WAs 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 WA. We defined WMRs as those refactorings that can be applied to the navigation and presentation models of a WA, with the purpose of improving its external characteristics, while preserving its behavior. We showed how to incorporate a quality evaluation method in the process in order to assess the improvement gained by refactoring. We also presented a case study showing how a typical shopping cart in an e-commerce site can improve its usability by applying some WMRs from our catalog.

Our current line of research is being devoted to extend our catalogue of WMRs and their possible composition, and to map each of the refactorings to quality attributes of a WA. A further research issue is to develop tool support both for applying WMRs and for enabling assessment, based on the OOHDM design method.

References

1. Beck, K.: Test-Driven Development. Addison-Wesley, Reading (2002)
2. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): a Modeling Language for Designing Web Sites. In: Proc. of WWW9, Amsterdam, NL (2000)

3. Fowler, M.: Refactoring: Improving the Design of Existing Code. Addison-Wesley, Reading (2000)
4. Garrido, A., Rossi, G., Distante, D.: Model Refactoring in Web Applications. In: WSE 2007. 9th IEEE Int'l Symposium on Web Site Evolution, Paris, France, IEEE CS Press, Los Alamitos (2007)
5. Ivkovic, I., Kontogiannis, K.: A Framework for Software Architecture Refactoring using Model Transformations and Semantic Annotations. In: Proc. IEEE, CSMR 2006, pp. 135–144 (2006)
6. Kerievsky, J.: Refactoring to Patterns. Addison-Wesley, Reading (2005)
7. Koch, N., Kraus, A.: The Expressive Power of UML-based Web Engineering. In: Magnusson, B. (ed.) ECOOP 2002. LNCS, vol. 2374, pp. 105–119. Springer, Heidelberg (2002)
8. Olsina, L., Rossi, G.: Measuring Web Application Quality with WebQEM. IEEE Multimedia 9(4), 20–29 (2002)
9. Olsina, L., Covella, G., Rossi, G.: Web Quality (Chapter fourth). In: Mendes, E., Mosley, N. (eds.) Springer Book titled "Web Engineering" (2006) ISBN 3-540-28196-7
10. Olsina, L., Papa, F., Molina, H.: How to Measure and Evaluate Web Applications in a Consistent Way (Chapter Thirteenth). In: Rossi, Pastor, Schwabe, Olsina (eds.) Springer Book titled Web Engineering: Modelling and Implementing Web Applications. Human-Computer Interaction Series, vol. 12 (2007) ISBN: 978-1-84628-922-4
11. Opdyke, W.: Refactoring Object-Oriented Frameworks. Ph.D.Thesis, Illinois University at Urbana-Champaign (1992)
12. Ping, Y., Kontogiannis, K.: Refactoring Web sites to the Controller-Centric Architecture. In: CSMR 2004, Tampere, Finland (2004)
13. Ricca, F., Tonella, P.: Program Transformations for Web Application Restructuring. In: Suh, W. (ed.) Web Engineering: Principles and Techniques. ch. 11, pp. 242–260 (2005)
14. Ricca, F., Tonella, P., Baxter, I.D.: Restructuring Web Applications via Transformation Rules. In: SCAM, pp. 152–162 (2001)
15. Roberts, D.: Eliminating Analysis in Refactoring. Ph.D. Thesis, Illinois University (1999)
16. Schwabe, D., Rossi, G.: An Object Oriented Approach to Web-Based Application Design. Theory and Practice of Object Systems 4(4) (1998)
17. UWA Consortium: Ubiquitous Web Applications. Proceedings of the eBusiness and eWork Conference e2002: Prague, Czech Republic (2002)
18. Van Duyne, D., Landay, J., Hong, J.: The Design of Sites. Addison-Wesley, Reading (2003)
19. Van Gorp, P., Stenten, H., Mens, T., Demeyer, S.: Towards automating source-consistent UML Refactorings. In: Proc. of the 6th Int'l Conference on UML (2003)
20. Zhang, J., Lin, Y., Gray, J.: Generic and Domain-Specific Model Refactoring using a Model Transformation Engine. In: Beydeda, S., Book, M., Gruhn, V. (eds.) Model-driven Software Development. ch. 9, pp. 199–218. Springer, Heidelberg (2005)