# A Web Usability Evaluation Process for Model-Driven Web Development

Adrian Fernandez, Silvia Abrahão, and Emilio Insfran

ISSI Research Group, Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, Camino de Vera, s/n, 46022, Valencia, Spain
{afernandez,einsfran,sabrahao}@dsic.upv.es

**Abstract.** Most of usability evaluation methods for the Web domain have several limitations such as: the concept of usability is only partially supported; usability evaluations are mainly performed when the Web application has been completed; there is a lack of guidelines on how to properly integrate usability into Web development. This paper addresses these issues through the presentation of a Web Usability Evaluation Process (WUEP) for integrating usability evaluations at different stages of model-driven Web development processes. A case study was performed in order to analyze the feasibility of the approach by applying WUEP to evaluate the usability of a real Web application that was developed by using a specific model-driven development process in the industrial domain. The results suggest that WUEP provides good insights into the performance of usability evaluations. The usability problems are corrected at the model and model-transformation levels, thereby improving the usability of the final Web application.

**Keywords:** Web Usability, Evaluation Process, Web Metrics, Model-Driven Web Development.

## 1 Introduction

Usability is considered to be one of the most important quality factors for Web applications, along with others such as reliability and security [23]. It is not sufficient to satisfy the functional requirements of a Web application in order to ensure its success. The ease or difficulty experienced by users of these Web applications is largely responsible for determining their success or failure. Usability evaluations and technologies that support the usability design process have therefore become critical in ensuring the success of Web applications [20].

The challenge of developing more usable Web applications has promoted the emergence of a large number of usability evaluation methods. However, most of these methods only consider usability evaluations during the last stages of the Web development process. Works such as Juristo *et al.* [17] claim that usability evaluations should also be performed during the early stages of the Web development process in order to improve user experience and decrease maintenance costs.

This is in line with the results of a systematic mapping study that we performed to investigate which usability evaluation methods have been used to evaluate Web

H. Mouratidis and C. Rolland (Eds.): CAiSE 2011, LNCS 6741, pp. 108–122, 2011.

artifacts and how they were employed [12]. The study suggests several areas for further research, such as the need for usability evaluation methods that can be applied during the early stages of the Web development process, methods that evaluate different usability aspects depending on the underlying definition of the usability concept, the need for evaluation methods that provide explicit feedback or suggestions to improve Web artifacts created during the process, and guidance for Web developers on how the usability evaluation methods can properly be integrated at relevant points of a Web development process.

The majority of Web development processes do not take advantage of the artifacts produced at the requirements and design stages. These intermediate artifacts are principally used to guide developers and to document the Web application. Since the traceability between artifacts and the final Web application is not well-defined, performing usability evaluations of these artifacts can be difficult. This problem is alleviated in Model-Driven Web Development processes (MDWD) where intermediate artifacts (models), which represent different views of a Web application, are used in all the steps of the development process, and the final source code is automatically generated from these models.

Most MDWD processes break up the Web application design into three models: content, navigation and presentation. These dimensions allow proper levels of abstraction to be established [7]. An MDWD process basically transforms models that are independent of technological implementation details (i.e., Platform-Independent Models - PIMs) such as structural models, navigational models or abstract user interface (UI) models into other models that contain specific aspects from a specific technological platform (i.e., Platform-Specific Models - PSMs) such as specific UI models, database schemas. This is done by automatically applying transformation rules. PSMs can be automatically compiled to generate the source code of the final Web application (Code Model - CM). This approach is followed by several methods such as: OO-H [13] or WebML [8]. The evaluation of these models (PIMs, PSMs, and CMs) can provide usability evaluation reports which propose changes that can be directly reflected in the final source code.

In a previous work [11] we followed these ideas to present a Web Usability Model that decomposes the usability concept into sub-characteristics and measurable attributes, which are then associated with Web metrics in order to quantify them. The aim was to evaluate usability attributes in several artifacts obtained from a Web development process that follows an MDWD approach. In this paper, we present an inspection method called Web Usability Evaluation Process (WUEP) that employs our Web Usability Model in order to integrate usability evaluations into several stages of MDWD processes.

This paper is organized as follows. Section 2 discusses related works that report usability evaluation processes based on inspection methods for Web development. Section 3 presents the Web Usability Evaluation Process. Section 4 presents a real case study that has been performed to illustrate the feasibility of WUEP. Finally, Section 5 presents our conclusions and further work.

## 2 Related Work

Usability evaluation methods can be mainly classified into two groups: empirical methods and inspection methods. Empirical methods are based on capturing and

analyzing usage data from real end-users, while inspection methods are performed by expert evaluators or designers and are based on reviewing the usability aspects of Web artifacts, which are commonly user interfaces, with regard to their conformance with a set of guidelines.

Usability inspection methods have emerged as an alternative to empirical methods as a means to identify usability problems since they do not require end-user participation and they can be employed during the early stages of the Web development process [9]. There are several proposals based on inspection methods to deal with Web usability issues, such as the Cognitive Walkthrough for the Web (CWW) [4] and the Web Design Perspectives (WDP) [10]. CWW assesses the ease with which a user can explore a Website by using semantic algorithms. However, this method only supports ease of navigation. WDP extends and adapts the heuristics proposed by Nielsen [21] with the aim of drawing closer to the dimensions that characterize a Web application: content, structure, navigation and presentation. However, this kind of methods tends to present a considerable degree of subjectivity in usability evaluations.

Other works present Web usability inspection methods that are based on applying metrics in order to minimize the subjectivity of the evaluation, such as the WebTango methodology [14] and Web Quality Evaluation Method (WebQEM) [22]. The WebTango methodology allows us to obtain quantitative measures, which are based on empirically validated metrics for user interfaces, to build predictive models in order to evaluate other user interfaces. WebQEM performs a quantitative evaluation of the usability aspects proposed in the ISO 9126-1 standard [15], and these quantitative measures are aggregated in order to provide usability indicators.

The aforementioned inspection methods are oriented towards application in the traditional Web development context; they are therefore principally employed in the later stages of Web development processes. As mentioned above, model-driven Web development offers a suitable context for early usability evaluations since it allows models, which are applied in all the stages, to be evaluated. This research line has emerged recently, and only a few works address Web usability issues, such as those of Atterer [3], Abrahão and Insfran [2], and Molina and Toval [19].

Atterer [3] proposed a prototype of a model-based usability validator with which to analyze models that represent enriched Web user interfaces. This approach takes advantage of models that represent the navigation (how the Website is traversed), and the UI of a Web application (abstract properties of the page layout).

Abrahão and Insfran [2] proposed a usability model to evaluate software products that are obtained as a result of model-driven development processes. Although this model is based on the usability sub-characteristics proposed in the ISO 9126 standard [15], it is not specific to Web applications and does not provide specific metrics. The same model was used in [24] with the aim of providing metrics for a set of attributes which would be applicable to the conceptual models that are obtained as a result of a specific MDWD process.

Molina and Toval [19] presented an approach to extend the expressivity of models that represent the navigation of Web applications in order to incorporate usability requirements. It improves the application of metrics and indicators to these models.

Nevertheless, to the best of our knowledge, there is no generic process for integrating usability evaluations into Model-Driven Web development processes.

## 3 Web Usability Evaluation Process

The Web Usability Evaluation Process (WUEP) has been defined by extending and refining the quality evaluation process that is proposed in the ISO 25000 standard [16]. The aim is to integrate usability evaluations into model-driven Web development processes by employing a Web Usability Model as the principal input artifact. This model decomposes the usability concept into sub-characteristics and measurable attributes, which are then associated with metrics in order to quantify them. These metrics provide a generic definition, which should be operationalized in order to be applicable to artifacts from different abstraction levels (PIMs, PSMs, and, CMs) in different MDWD processes (e.g., OO-H, WebML, UWE).

Figure 1 shows an overview of the main stages of WUEP. Three roles are involved: evaluation designer, evaluator, and Web developer. The *evaluation designer* performs the first three stages: 1) Establishing the requirements of the evaluation, 2) Specification of the evaluation, and 3) Design of the evaluation. The *evaluator* performs the fourth stage: 4) Execution of the evaluation. Finally, the *Web developer* performs the last stage: 5) Analysis of changes. The following sub-sections describe each of the main stages by including the activities into which they are decomposed.

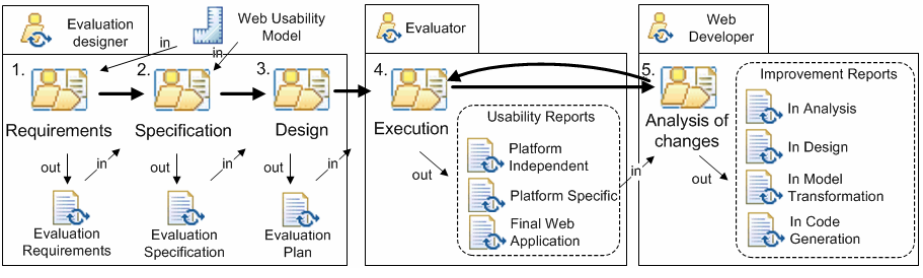


**Fig. 1.** An overview of the Web Usability Evaluation Process

### 3.1 Establishment of Evaluation Requirements

The aim of this stage is to establish the requirements of the evaluation to delimit the scope of the evaluation. The activities involved in this stage are described below.

**1. Establish the Purpose of Evaluation.** This activity determines the aim of the usability evaluation, i.e., whether the evaluation will be performed in a Web application in order to provide feedback during the Web development process, or whether different Web applications, belonging to the same Web application family, are compared in order to establish a ranking among them.

**2. Specify Profiles.** The different factors that will condition the evaluation are determined. These factors are: the *type of Web application*, since each family of Web applications has different goals that make an impact on the selection of usability attributes, i.e., navigability might be more relevant to Intranets whereas attractiveness might be more relevant to social networks; the *Web development method*, since knowledge about its process and artifacts is needed in order to properly integrate the

usability evaluations; and the *context of use*, which takes into account parameters such as type of users, user age, work environment, etc.

**3. Select the Web Artifacts to Be Evaluated.** The artifacts selected may depend on either the Web development method or the technological platform. The artifacts to be considered might be: *Platform-Independent models* (e.g., content/domain models, navigational models, abstract user interface models) which are obtained as output from the analysis and design stages of an MDWD process; *Platform-Specific models* (e.g., specific user interface models, database schemas) which are obtained as output from the model transformation stage of an MDWD process; and *Code models* (e.g., source code, final user interfaces) which are obtained as output from the code generation stage of an MDWD process.

**4. Select Usability Attributes.** The Web Usability Model is used as a catalog in order to select which usability attributes will be evaluated. This Web Usability Model is based on the decomposition of the usability characteristic proposed in the ISO 25000 (SQuaRE) [16] quality model. The first version of this model was presented in [11] and has been improved by considering other usability guidelines (e.g., [18]), in order to discover 15 new usability attributes that are relevant to the Web domain. The Web Usability Model currently considers two different views: usability of the software product, and usability in use. In this paper, we focus solely on the usability of the software product, since it can be assessed during the Web development process by inspecting Web artifacts. Usability from a software product perspective is decomposed into seven sub-characteristics: *Appropriateness recognisability*, *Learnability*, *Ease of use*, *Helpfulness*, *Technical accessibility*, *Attractiveness*, and *Compliance*. These are also decomposed into other sub-characteristics and measurable attributes. The Web Usability Model, including all the sub-characteristics attributes and their associated metrics, is available at `http://www.dsic.upv.es/~afernandez/CAiSE11/WebUsabilityModel`.

The outcomes of the above activities represent the *Evaluation Requirements* that will be used as input by the next stage.

### 3.2 Specification of the Evaluation

The aim of this stage is to specify the evaluation in terms of which metrics are intended to be applied and how the values obtained by these metrics allow usability problems to be detected. The activities involved in this stage are described below.

**1. Select the Metrics to Be Applied.** The Web Usability Model is used to discover which of the metrics are associated with the usability attributes selected. Metrics allow us to interpret whether or not these attributes contribute to achieving a certain degree of usability in the Web application. The Web metrics that are included in the Web Usability Model were taken from different sources: surveys that contain metrics that had been theoretically and empirically validated (e.g., Calero *et al.* [6]); quality standards (e.g., SQuaRE [16]) and Web guidelines (e.g., W3C [26]). Each metric was studied by considering its parameters (i.e., purpose, interpretation, artifacts to which it can be applied, etc.) in order to provide a generic definition of the metric. The aim of providing a generic definition is to allow metrics to be applied to artifacts of different

abstraction levels and from different MDWD processes. Appendix A includes two examples of metrics from the Web Usability Model with their generic definition.

**2. Operationalize the Metrics.** The calculation formulas of the selected metrics should be operationalized by identifying variables from the generic definition of the metric in the modeling primitives of the selected artifacts, in other words, by establishing a mapping between the generic description of the metric and the concepts that are represented in the artifacts. In the evaluation of models (PIM, PSM, and CM), the calculation of the operationalized formulas may require assistance from an evaluator to determine the values of the variables involved, or it may require a verification tool if these formulas are expressed in variables that can be automatically computed from the input models by query languages such as the Object Constraint Language (OCL).

**3. Establish Rating Levels for Metrics.** Rating levels are established for ranges of values obtained for each metric by considering their scale type and the guidelines related to each metric whenever possible. These rating levels allow us to discover whether the associated attribute improves the Web application's level of usability, and are also relevant in detecting usability problems that can be classified by their level of severity.

The outcomes of the above activities represent the *Evaluation specification* that will be used as input by the next stage.

### 3.3 Design of the Evaluation

The aim of this stage is to design how the evaluation will be performed and what information will be collected during the evaluation.

**1. Define the Template for Usability Reports.** This template is defined in order to present all the data that is related to the usability problems detected. A usability report is commonly a list of usability problems (UP). Each UP can be described by the following fields: *ID*, which refers to a single UP; *description* of the UP; affected attribute from the Web Usability Model; *severity level*, which could be low, medium or critical; *artifact evaluated*, in which metrics have been applied; *source of the problem*, which refers to the artifact that originates the usability problem (e.g., PIMs, PSMs, CMs, and transformation rules); *occurrences*, which refer to the number of appearances of the same UP; and *recommendations* to correct the UP detected (some recommendations might also be automatically provided by interpreting the range values). Other fields that are useful to post-analyze the UP detected can also be added, such as *priority* of the UP; *effort* that is needed to correct the UP; and *changes* that must be performed in order to take the aforementioned fields into consideration.

**2. Elaborate an Evaluation Plan.** Designing the evaluation plan implies: establishing an evaluation order of artifacts; establishing a number of evaluators; assigning tasks to these evaluators, and considering any restrictions that might conditioned the evaluation. The recommended order is to first evaluate the artifacts that belong to a higher abstraction level (PIMs), since these artifacts drive the development of the final Web application. This allows us to detect usability problems during the early stages of the Web development process. The artifacts that belong to a lower level of abstraction (PSMs and CMs) are then evaluated.

The outcomes of the above activities represent the *Evaluation Plan* that will be used as input by the next stage.

### 3.4 Execution of the Evaluation

The aim of this stage is to execute the evaluation in accordance with the Evaluation Plan. The evaluator applies the operationalized metrics to the artifacts that have been selected. If the rating levels obtained identify a UP, the elements of the artifact involved that contribute to achieving this metric value are analyzed. This helps us to determine the source of the usability problem thanks to the traceability that exists among the models in an MDWD process.

The outcomes of this stage are: a *platform-independent usability report*, which collect the UPs that are detected during the evaluation of PIMs; a *platform-specific usability report*, which collects the UPs that are detected during the evaluation of PSMs; and a *final Web application usability report*, which collects the UPs that are detected during the evaluation of CMs.

### 3.5 Analysis of Changes

The aim of this stage is to classify all the UPs detected from each of the usability reports shown above and to analyze the recommendations provided in order to propose changes with which to correct the artifacts. Usability problems whose source is located in PIMs that are related to content and navigation (e.g., UP detected in structural models, navigational models) are collected to create the *improvement report in analysis*. Usability problems whose source is located in PIMs that are related to presentation (e.g., UP detected in abstract user interfaces models) are collected to create the *improvement report in design*. Usability problems whose source is located in PSMs or transformation rules among PIMs and PSMs are collected to create the *improvement report in model transformation*. Finally, usability problems whose source is located in the generation rules among PSMs and CMs are collected to create the *improvement report in code generation*. The last two reports are useful for providing feedback in order to improve the Computer-Aided Web Engineering tool (CAWE) that supports the Web development method and performs the transformations among models, along with the generation rules among models and the final source code.

It is important to note that after applying the changes suggested by the improvement reports, re-evaluations of the artifacts might be necessary.

## 4 Case Study

An exploratory case study was performed by following the guidelines presented in [25] in order to study the feasibility of applying WUEP in industrial contexts. The stages of which the case study was comprised were: design, preparation, collection of data, and analysis of data, each of which is explained below.

### 4.1 Design of the Case Study

The case study was designed by considering the five components that are proposed in [25]: purpose of the study, underlying conceptual framework, research questions to be addressed, sampling strategy, and methods employed.

The purpose of the case study is to show the feasibility of applying WUEP to discover usability problems at different levels of abstraction of a MDWD process. The conceptual framework that links the phenomena to be studied is the idea based on integrating usability evaluations into MDWD processes [11]. The research questions that are intended to be addressed are: *a) What type of usability problems can be detected in each phase of a model-driven Web development process and what are their implications for intermediate artifacts (models)?*, and *b) What limitations does the Web Usability Evaluation Process present?*.

The sampling strategy of the case study is based on an embedded single-case design. We contacted a Web development company located in Alicante (Spain) in order to apply WUEP to a real Web application from a project in progress. The company provided us with access to a task management system. The aim was to perform evaluations on the artifacts that define this Web application at different abstraction levels in an MDWD process.

WUEP was applied as follows: two of the authors performed the evaluation designer role in order to design an evaluation plan (in critical activities such as the selection of usability attributes, we required the help of two external Web usability experts); and the evaluator role was performed by five evaluators: the other author and four other independent evaluators with an average of five years' experience in usability evaluations. The technique that was used to obtain feedback about the feasibility of WUEP was the analysis of the usability reports and observation of the evaluators when performing the execution of the evaluation. We have not considered the *analysis of changes* stage since the aim of this case study was to show the feasibility of WUEP in discovering usability problems.

### 4.2 Preparation of the Case Study

An evaluation plan was defined by following WUEP. With regard to the *establishment of evaluation requirements* stage of WUEP, the purpose of the evaluation was to perform a usability evaluation during the development of the Web application mentioned above. The type of Web application was an Intranet that was developed using the OO-H method [13] supported by the VisualWade tool (`www.visualwade.com`). The context in which the Web application will be used is a software development company, and there are two kinds of users: project manager and programmers.

OO-H is an MDWD method that provides the semantics and notation for developing Web applications. The platform-independent models (PIMs) that represent the different concerns of a Web application are: a class model, a navigational model, and a presentation model. The class model is UML-based and specifies the content requirements; the navigational model is composed of a set of Navigational Access Diagrams (NADs) that specify the functional requirements in terms of navigational needs and users' actions; and the presentation model is composed of a set of Abstract Presentation Diagrams (APDs), whose initial version is obtained by merging the former models, which are then refined in order to represent the visual properties of the final UI. The platform-specific models (PSMs) are embedded into a model compiler, which automatically obtains the source code (CM) from the Web application by taking all the previous PIMs as input.

The Web artifacts selected to be evaluated were all the platform-independent models and the final UIs (i.e., 1 class model, 4 NADs, 4 APDs and 4 final UIs). PSMs cannot be evaluated since they are embedded in the model compiler. Figure 2(a) shows an excerpt of a NAD and Figure 2(b) shows an excerpt of an APD.

A set of 20 usability attributes were selected from the Web Usability Model through the consensus of the evaluator designers and the Web usability experts. The attributes were selected by considering which of them would be more relevant to the type of Web application and the context in which it is going to be used (e.g., Interconnectivity, Clickability, Permanence of links, etc.). Only 20 attributes were selected in order to avoid extending the duration of the execution stage.

With regard to the *specification of the evaluation* stage of WUEP, metrics associated with the selected attributes were obtained from the Web Usability Model, and then associated with the artifact in which they could be applied. Since metrics can be applied at different abstraction levels, the highest level of application was selected. For example, the metrics presented in Appendix A indicate that: the Compactness metric could be applied to models that specify the navigation (i.e., NADs); and the Discernible links metric could be applied to models that specify the UI (i.e., APDs).

Once the metrics had been associated with the artifacts, metrics were operationalized in order to provide a calculation formula for artifacts from the OO-H method (in this case), and to establish rating levels for them. Appendix B shows an example of operationalization and rating levels for the metrics from Appendix A.

With regard to the *design of the evaluation* stage of WUEP, a template for usability reports was defined by considering the same fields proposed in the previous section. The evaluation plan that was elaborated takes into consideration the following evaluation order of the artifacts: class model; NADs, APDs, and final UIs. Five evaluators were selected to perform the execution stage of WUEP, as mentioned above. This number was selected by following the recommendations of studies which claim that five or more evaluators are needed to perform usability inspections [21]. The evaluators attended a training session of 2 hours at which they were informed of the artifacts from the OO-H method and the tasks to be performed.

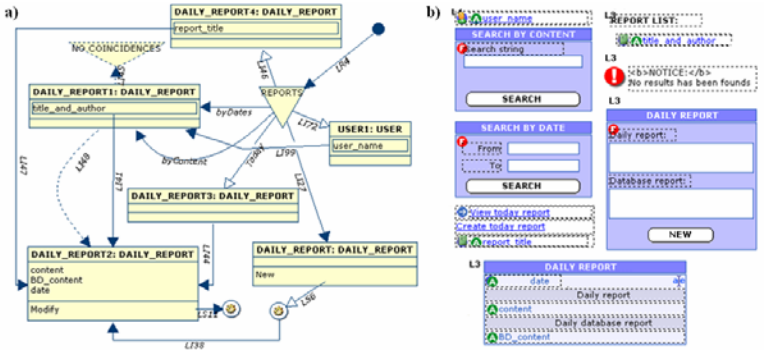


**Fig. 2.** Excerpts from artifacts (PIMs) of the Web application developed in OO-H

### 4.3 Collection of the Data

The data for this case study was collected during the *execution of the evaluation* stage of WUEP. As an example, we show values obtained after the evaluators applied certain operationalized metrics to the excerpts of models that are shown in Figure 2.

In the NAD (Fig. 2 (a)), the value of the Compactness metric is: Cp (NAD) = (448-347) / (448-56) = 0.26 since the components of the formula are: n=k=8; Max=448; Min=56; and $\sum i \sum j$ Dij=347. The rating level of this metric ($0.2 < x \leq 0.8$) therefore indicates that there is no usability problem related to the Interconnectivity attribute in this NAD.

In the APD (Fig. 2 (b)), the value of the Discernible links metric is: DL (APD) = 1-0.5(2/5+3/13) = 0.68 since there are 2 links (*title_and_author* and *report_title*) that might be confused with labels, and there are 3 labels (*date*, *content* and *BD_content*) that might be confused with links as a result of their visual properties. The rating level of this metric ($0.5 < x \leq 0.7$) therefore indicates the existence of a medium usability problem (UP001) related to the Clickability attribute in this APD.

Each evaluator presented the usability problems detected in usability reports by using the defined template. As an example, Table 1 shows an excerpt of the usability report that is presented in the UP001.

**Table 1.** Usability problem detected: UP001

| ID | UP001 |
|---|---|
| Description | *title_and_author* and *report_title* links are confused with labels, whereas *date*, *content* and *BD_content* labels are confused with links. |
| Affected attribute | Appropriateness recognisability / Navigability / Clickability |
| Severity level | Medium (0.68) |
| Artifact evaluated | Abstract Presentation Diagram (APD no. 3) |
| Source problem | Abstract Presentation Diagram (APD no. 3) |
| Occurrences | 5 errors in the same APD |
| Recommendations | Change the visual properties of links and labels to another face/color in order for them to be discernible. The use of typographies, such as blue, and an underlined face for links is recommended. |

### 4.4 Analysis of Data

The usability problems reported in the execution stage were analyzed to address our research questions.

**Usability Problems and their Implications for Intermediate Artifacts.** The evaluation performed for the case study only considers two different abstraction levels: the platform-independent models: class model, navigational model (NADs), and presentation model (APDs); and the code model: final user interfaces.

With regard to the class model, a mean of 0.6 UPs (std. dev. 0.54) were detected by the evaluators since the expressiveness of its modeling primitives makes it difficult to operationalize metrics. However, UPs related to the minimization of user effort were detected (e.g., *"does not provide default values for elements that are needed for user input"*). These can be solved by correcting the class model itself by adding default values to attributes from classes.

With regard to the navigational model, a mean of 2.2 UPs (std. dev. 0.83) were detected by the evaluators since the navigation steps for carrying out possible actions are considered to be proper by values of metrics such as navigation density, navigation depth, etc. However, UPs related to the lack of return paths were detected (e.g., *"does not provide backs link"* or *"no link to go home"*). These can be solved in the navigational model itself by adding links that connect the information previously obtained.

With regard to the presentation model, a mean of 10.6 UPs (std. dev. 1.51) were detected by the evaluators since the expressiveness of its modeling primitives allows several metrics to be operationalized (e.g., *"error messages are not meaningful"* and *"links and labels are not discernible"*). The former can be solved in the presentation model itself by correcting the label that shows the message, and the latter can be solved in the navigational model since names for links can be provided as properties of them.

With regard to the final user interface, a mean of 12.2 UPs (std. dev. 1.78) were detected by the evaluators. Some examples of the UPs detected were related to the support to operation cancellation (e.g., "*the creation of a new task cannot be canceled"*). This can be solved in the navigational model since links can be added to permit cancelation paths. Other UPs related to the immediate feedback that provide UI controls were also detected (e.g., *"tabs of the main menu do not show the current user state"*). This can be solved in the transformation rules that map the representation of tabs with a specific UI control of the technological platform.

It is important to note that some usability attributes such as the uniformity of the UI position are directly supported by the modeling primitives of the APDs.

**Lessons Learned.** The case study has been useful in that it has allowed us to learn more about the potentialities and limitations of our proposal and how WUEP can be improved.

WUEP can detect several usability problems from a wide range of types in several artifacts employed during the early stages of an MDWD process. The application of operationalized metrics and traceability among models not only provides a list of usability problems, but also facilitates the provision of recommendations with which to correct them. Metric operationalization also allows WUEP to be applied to different MDWD processes by establishing a mapping between the generic description of the metric and the modeling primitives of artifacts, in addition to other traditional development processes, by operationalizing metrics only in final user interfaces. However, usability reports will only provide feedback to the implementation stage since traceability between artifacts is not well-defined. Moreover, the evaluation process may be a means to discover which usability attributes are directly supported by the modeling primitives or to discover limitations in the expressiveness of these artifacts.

During the execution of the case study, several aspects related to how WUEP is applied were detected to be improved. For example, the application of metrics was detected as being a very tedious task when performed manually, particularly when the metrics provided a complex calculation formula (e.g., Compactness, Depth of the navigation, etc.). In addition, some metrics obtained different values depending on the evaluator since these metrics imply a certain degree of subjectivity (e.g., Discernible links, properly chosen Metaphors, etc.). These issues could be alleviated, at least to some extent, by providing better guidelines in order to minimize the subjectivity in

the employment of operationalized metrics and by developing a tool that automates a large part of the evaluation process.

**Limitations of the Study.** Since the aim of the case study was to show the feasibility of applying WUEP in an MDWD process in industrial contexts, we selected a real Web application that was under development by a company, and we focused principally on the first four stages of WUEP. However, this case study presents some limitations, such as the fact that only one type of Web application was considered since the company imposed certain restrictions. Although the usability attributes to be evaluated were selected through the consensus of the evaluation designers and the two independent Web usability experts, they might not have been very representative. In order to overcome such limitations it is necessary to determine which usability attributes are most relevant for each family of Web applications by considering several opinions from Web usability experts.

## 5 Conclusions and Further Work

This paper presents the Web Usability Evaluation Process (WUEP) as a usability inspection method that integrates usability evaluations during several stages of Model-Driven Web Development (MDWD) processes. WUEP provides broad support to the concept of usability since its underlying Web Usability Model has been extended and adapted to the Web domain by considering the new ISO 25000 series of standards (SQuaRE), along with several usability guidelines. The explicit definition of the activities and artifacts of WUEP also provides evaluators with more guidance and offers the possibility of automating (at least to some extent) several activities in the evaluation process by means of a process automation tool.

We believe that the inherent features of MDWD processes (e.g., traceability between models by means of model transformations) provide a suitable environment for performing usability evaluations. The integration of WUEP into these environments is thus based on the evaluation of artifacts, particularly intermediate artifacts (models), at several abstraction levels from different MDWD processes. The evaluation of these models (by considering the traceability among them) allows the source of the usability problem to be discovered and facilitates the provision of recommendations to correct these problems during the earlier stages of the Web development process. This signifies that if the usability of an automatically generated user interface can be assessed, the usability of any future user interface produced by MDWD processes could be predicted. In other words, we are referring to a user interface that can be usable by construction [1], at least to some extent. Usability can thus be taken into consideration throughout the entire Web development process. This enables better quality Web applications to be developed, thereby reducing effort at the maintenance stage.

In the future we intend to do the following: to analyze different proposals concerning the inclusion of aggregation mechanisms to merge values from metrics in order to provide scores for usability attributes that will allow different Web applications from the same family to be compared; to determine the most relevant usability attributes for different families of Web applications according to Web domain experts in order to provide pre-defined selections of operationalized metrics; to apply WUEP to different MDWD processes; to evaluate different types of Web applications such as Web

applications 2.0 and Rich Internet Applications (RIA) that are developed thorough an MDWD process; to perform controlled experiments in order to assess the effectiveness and ease of use of WUEP by comparing it to other usability evaluation methods; and finally, to develop a tool with the capability of automating a large part of the evaluation process.

**Acknowledgments.** This research work is funded by the MULTIPLE project (TIN2009-13838) and the FPU program (AP2007-03731) from the Spanish Ministry of Science and Education.

## References

1. Abrahão, S., Iborra, E., Vanderdonckt, J.: Usability Evaluation of User Interfaces Generated with a Model-Driven Architecture Tool. In: Maturing Usability: Quality in Software, Interaction and Value, Springer, pp. 3-32 (2007)
2. Abrahão, S., Insfran, E.: Early Usability Evaluation in Model-Driven Architecture Environments. In: 6th IEEE International Conference on Quality Software (QSIC 2006), pp. 287–294. IEEE Computer Society, Beijing (2006)
3. Atterer, R., Schmidt, A.: Adding Usability to Web Engineering Models and Tools. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 36–41. Springer, Heidelberg (2005)
4. Blackmon, M.H., Polson, P.G., Kitajima, M., Lewis, C.: Cognitive Walkthrough for the Web. In: Proc. of the ACM CHI 2002, USA, pp. 463–470 (2002)
5. Botafogo, R., Rivlin, E., Shneiderman, B.: Structural analysis of hypertexts: Identifying hierarchies and useful metrics. ACM Trans. Inf. Systems 10(2), 142–180 (1992)
6. Calero, C., Ruiz, J., Piattini, M.: Classifying Web metrics using the Web quality model. Online Information Review Journal. Emerald Group. 29(3), 227–248 (2005)
7. Casteleyn, S., Daniel, F., Dolog, P., Matera, M.: Engineering Web Applications. Springer, Heidelberg (2009)
8. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. In: Proc. of the 9th WWW Conf., pp. 137–157 (2000)
9. Cockton, G., Lavery, D., Woolrychn, A.: Inspection-based evaluations. In: Jacko, J.A., Sears, A. (eds.) The Human-Computer Interaction Handbook, 2nd edn., pp. 1171–1190 (2003)
10. Conte, T., Massollar, J., Mendes, E., Travassos, G.H.: Usability Evaluation Based on Web Design Perspectives. In: 1st Int. Symposium on Empirical Software Engineering and Measurement (ESEM 2007), Spain, pp. 146–155 (2007)
11. Fernandez, A., Insfran, E., Abrahão, S.: Integrating a Usability Model into Model-Driven Web Development Processes. In: Vossen, G., Long, D.D.E., Yu, J.X. (eds.) WISE 2009. LNCS, vol. 5802, pp. 497–510. Springer, Heidelberg (2009)
12. Fernandez, A., Insfran, E., Abrahão, S.: Usability Evaluation Methods for the Web: A Systematic Mapping Study. In: Information and Software Technology (2011), doi:10.1016/j.infsof.2011.02.007
13. Gomez, J., Cachero, C., Pastor, O.: Conceptual Modeling of Device-Independent Web Applications. IEEE MultiMedia 8(2), 26–39 (2001)
14. Ivory, M.Y.: An Empirical Foundation for Automated Web Interface Evaluation. PhD Thesis, University of California, Berkeley, Computer Science Division (2001)
15. ISO/IEC 9126-1 Standard, Software Engineering, Product Quality - Part 1: Quality Model (2001)
16. ISO/IEC 25000 series, Software Engineering, Software Product Quality Requirements and Evaluation, SQuaRE (2005)

17. Juristo, N., Moreno, A., Sanchez-Segura, M.I.: Guidelines for eliciting usability functionalities. IEEE Transactions on Software Engineering 33(11), 744–758 (2007)
18. Leavit, M., Shneiderman, B.: Research-Based Web Design & Usability Guidelines. U.S. Government Printing Office (2006), http://usability.gov/guidelines/index.html (last access: March 2011)
19. Molina, F., Toval, J.A.: Integrating usability requirements that can be evaluated in design time into Model Driven Engineering of Web Information Systems. Advances in Engineering Software 40(12), 1306–1317 (2009)
20. Neuwirth, C. M., Regli, S. H.: IEEE Internet Computing Special Issue on Usability and the Web, vol. 6(2) (2002)
21. Nielsen, J., Molich, R.: Heuristic evaluation of user interfaces. In: Proc. ACM CHI 1990 Conf., Seattle, WA, pp. 249–256 (1990)
22. Olsina, L., Rossi, G.: Measuring Web Application Quality with WebQEM. IEEE Multimedia 9(4), 20–29 (2002)
23. Offutt, J.: Quality Attributes of Web Software Applications. IEEE Software: Special Issue on Software Engineering of Internet Software, 25–32 (2002)
24. Panach, I., Condori, N., Valverde, F., Aquino, N., Pastor, O.: Understandability measurement in an early usability evaluation for model-driven development: an empirical study. In: Int. Empirical Software Engineering and Measurement (ESEM 2008), pp. 354–356 (2008)
25. Runeson, P., Höst, M.: Guidelines for Conducting and Reporting Case Study Research in Software Engineering. Empirical Software Engineering 14(2) (2009)
26. World Wide Web Consortium W3C: Web Content Accessibility Guidelines (WCAG) 2.0 (2008), http://www.w3.org/TR/WCAG/ (last access: March 2011)

## Appendix A: Examples of Metrics with Their Generic Description

| Name | Compactness (Cp) |
|---|---|
| Attribute | Appropriateness recognisability / Navigability / Interconnectivity. |
| Description | The degree of interconnection among nodes belonging to a hypermedia graph. Its formula is Cp = (Max – ∑i ∑j Dij) / (Max – Min), where Max = (n2 - n)*k; Min = (n2 - n); n = quantity of nodes in the graph; k = constant superior to the amount of nodes; ∑i ∑j Dij = the sum of distances taken from the matrix of converted distances (with factor k); and Dij = the distance between the nodes i and j. |
| Scale type | Ratio between 0 and 1. |
| Interpretation | Values closer to 1 signify that each node can easily reach any other node in the graph. However, a fully connected graph may lead to disorientation. |
| Application level | At the PIM/PSM level, if navigation is modeled as a graph where nodes represent pages and edges are links among pages; and at the CM level by analyzing the link's targets included in the source code for each page. |

| Name | Discernible links (DL) |
|---|---|
| Attribute | Appropriateness recognisability / Navigability / Clickability. |
| Description | The degree to which links are discernible from other textual elements. Its formula is DL = 1 - 0.5(CL / NL + CT / NT), where CL = the number of links that may be confused with textual elements; NL= total number of links; CT = number of textual elements that may be confused with links; and NT = total number of textual elements. |
| Scale type | Ratio between 0 and 1. |
| Interpretation | Values closer to 1 indicate that users can differentiate links from textual elements. |
| Application level | At the PIM/PSM level, if the expressiveness of abstract user interface models allows aesthetic properties to be defined for links such as color or style; and at the CM level by assessing the elements from the source code that define the visual appearance of links in the final user interface (e.g., Cascading Style Sheets). |

## Appendix B: Examples of Operationalized Metrics for OO-H

*Compactness (Cp)*: In an OO-H navigational model, the compactness is calculated by considering the navigational classes and links of each NAD as a hypermedia graph in which classes are nodes and links are edges. If we consider the scale type and the recommendations suggested by Botafogo *et al.* [5], then the rating levels are established as no UP ($0.2 < x \leq 0.8$), and medium UP ($0 \leq x < 0.2$ and $0.8 < x \leq 1$).

*Discernible links (DL)*: In an OO-H presentation model, the visual properties of textual elements in each APD (i.e., links and labels) are represented as attributes such as font, color, and face. The operationalized metric is:

$$DL\ (APD) = 1 - 0.5(CLi / NTLi + CLa / NTLa)$$

where *CLi* signifies the number of links with similar visual properties to most of the labels, *CLa* signifies the number of labels with similar visual properties to most of the links, whereas *NTLi* and *NTLa* signify the total number of links and labels, respectively. If we consider the scale type, then the rating levels are established as no UP ($0.99 < x \leq 1$), low UP ($0.7 < x \leq 0.99$), medium UP ($0.5 < x \leq 0.7$), and critical UP ($0 < x \leq 0.5$).