Integrating a Usability Model into Model-Driven Web Development Processes

Adrian Fernandez, Emilio Insfran, and Silvia Abrahão

ISSI Research Group. Department of Information Systems and Computation, Universidad Politécnica de Valencia, Camino de Vera, s/n, 46022, Valencia, Spain
{afernandez,einsfran,sabrahao}@dsic.upv.es

Abstract. Usability evaluations should start early in the Web development process and occur repeatedly throughout all stages to ensure the quality of the Web application, not just when the product is completed. This paper presents a Web Usability Model, which is aligned with the SQuaRE standard, to evaluate usability at several stages of a Web development process that follows a Model-Driven Development (MDD) approach. The Web Usability Model is generic and must be operationalized into a concrete MDD method by specifying the relationships between the usability attributes of the Usability Model and the modeling primitives of the specific Web development method. To illustrate the feasibility of the approach, we present a case study where the Usability Model has been applied in the evaluation of the models that are produced during the Web application development process.

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

1 Introduction

Web applications have become the backbone of business and information exchange and the ease or difficulty that users experience with these systems will determine their success or failure. The challenge of developing more usable Web applications has led to emergence of a variety of techniques, methods, and tools to address Web usability.

Usability evaluations of Web applications can be performed by employing quality models since they define the term *usability* as a quality characteristic that can be decomposed into specific measurable attributes. However, most of the usability evaluation approaches [15], [18], [19] only consider usability evaluation at implementation stages when the product is completed.

As Web applications must be usable to be accepted by users, usability evaluations should start early in the Web development process and occur repeatedly throughout the design cycle, not just when the product is completed to avoid source code maintenance. Usability evaluation at each stage of the Web application development process is a critical part of ensuring that the product will actually be used and be effective for its intended purposes.

In a Model-Driven Web development process, models that specify an entire Web application are used in all steps of the process. This allows usability issues to be considered at early stages by evaluating the models that drive the implementation of a

G. Vossen, D.D.E. Long, and J.X. Yu (Eds.): WISE 2009, LNCS 5802, pp. 497–510, 2009.

final Web application [2]. A Web development process that follows a Model-Driven approach basically transforms models that are independent from implementation details (Platform-Independent Models - PIM) into other models that contain specific aspects from a concrete platform (Platform-Specific Models - PSM). PSM models can be compiled to automatically generate the Web application source code (Code Model - CM). Therefore, a better quality Web application can be obtained by evaluating and correcting models without requiring source code maintenance.

In this paper, we propose a Web Usability Model that is aligned with the ISO/IEC 25000 standard (SQuaRE) [13] in order to evaluate and improve usability at several stages of a Web development process that follows a Model-Driven Development (MDD) approach. The model is generic and must be operationalized into a concrete Web development method based on the MDD approach such as OO-H [8], UWE [14], or WebML [7]. Our strategy assesses intermediate artifacts (PIM and PSM models) as well as the final Web application source code (CM model) in order to take usability into account throughout the entire Web development process.

This paper is organized as follows. Section 2 discusses related work, in particular, previous quality and usability models for Web applications. Section 3 presents the strategy for integrating the Usability Model into a Model-Driven Web development process. Section 4 presents the Web Usability Model. Section 5 shows a case study which describes how the proposed model was applied to evaluate a real Web application that was developed following the OO-H method [8]. Finally, Section 6 presents the conclusions and further work.

2 Related Work

Several quality models for Web usability evaluation have been proposed in the last few years. These models have been defined from scratch or defined based on existing standards such as ISO/IEC 9241-11 [10] and ISO/IEC 9126-1 [9].

Quality models for the Web context that are defined from scratch include the proposals of Becker and Mottay [5], Sutcliffe [19], and Signore [18]. Becker and Mottay [5] present a usability assessment model to identify and measure usability factors. The factors defined are page layout, navigation, design consistency, information content, performance, customer service, reliability, and security. However, all these factors were measured at the final user interface (UI) of a Web application.

Sutcliffe [19] presents a model based on initial attractiveness, navigation and transaction. This work mainly focuses on how attractiveness can be operationalized in terms of design guidance. The attractiveness characteristic was divided into generic aspects of a final UI such as aesthetic design, use of media to direct attention, issues of linking visual styles, etc.

Signore [18] presents a quality model with a set of characteristics relating internal and external quality factors that can be measured by automated tools. The model distinguishes five dimensions related to *correctness* of the source code, *presentation* criteria (page layout, text presentation, etc.), *content* issues (readability, information structure, etc.), *navigation* aspects, and ease of *interaction* (transparency, recovery, help and hints, etc.).

On the other hand, the quality models for the Web context that are defined based on existing standards include the proposals of Olsina and Rossi [16], Calero *et al.* [6],

Seffah *et al.* [17], and Moraga *et al.* [15]. Olsina and Rossi [16] proposed the Web Quality Evaluation Method (WebQEM) to define quality characteristics and attributes based on the ISO/IEC 9126 such as *usability*, *functionality*, *reliability,* and *effectiveness*; the Web audience's needs are also incorporated. However, the evaluation of these quality characteristics takes place at the operational phases of the Web application. WebQEM is supported by a tool that relies on a Web-based hyper-document model that supports traceability of evaluation aspects.

Calero *et al.* [6] present the Web Quality Model (WQM), which distinguishes three dimensions: Web features (*content*, *presentation*, and *navigation*); quality characteristics based on the ISO/IEC 9126 (*functionality*, *reliability*, *usability*, *efficiency*, *portability*, and *maintainability*); and lifecycle processes from ISO/IEC 12207 [11] (*development*, *operation* and *maintenance*) including organizational processes such as *project management* and reuse *program management*. WQM incorporates a total of 326 Web metrics (taken from the existing literature) which was classified according to these three dimensions.

Seffah *et al.* [17] present the Quality in Use Integrated Measurement (QUIM) as a consolidated model for usability measurement in Web applications. QUIM combines existing models from ISO/IEC 9126, ISO/IEC 9241-11, and others. It decomposes usability into factors, and then into criteria (a criterion can belong to different factors). Finally, these criteria are decomposed into specific metrics that can quantify the criteria. The QUIM model is supported by an editor tool that manages a repository of usability measurement plans and defines new metrics.

Moraga *et al.* [15] present a usability model oriented to portlets evaluation. Portlets are pluggable UI software components that are managed and displayed in a web portal. The portlet usability model is based on the sub-characteristics from ISO/IEC 9126 (*understandability*, *learnability*, and *compliance*); nevertheless, the *operability* sub-characteristic was replaced by *customizability*, which is closer to the portlet context. Measures are based on a number ranking with an acceptance threshold.

After reviewing several quality model approaches, we have identified a lack of quality models that can evaluate Web usability not only when the Web application is implemented, but also at earlier stages of development, such as the analysis and design stages. In a previous work, Abrahão and Insfran [3] proposed a usability model for early evaluation in Model-Driven Architecture environments. In this model, usability was decomposed into the same sub-characteristics as the ones in the ISO/IEC 9126 (*learnability*, *understandability*, *operability*, and *compliance*), and then decomposed again, into more detailed sub-characteristics and attributes. This last decomposition was performed taking into account a set of ergonomic criteria for UIs [4]. Relationships between the elements from the PIM/CM models of a specific MDD method and the usability attributes of the model were then established. However, the model was not proposed for the Web domain and it did not provide metrics for measuring the model attributes.

3 Integrating a Usability Model into the MDD Process

Recent studies indicate that the adoption of Model-Driven Development (MDD) has increased. Currently, there are several Web development methodologies that follow

this approach, such as OO-H [8], WebML [7], or UWE [14]. These methods support the development of a Web application by defining different views (models), including at least one structural model, a navigational model, and an abstract presentation model. Some methods also provide model transformations and automatic code generation.

The usability of a Web application obtained as a result of this transformation process can be assessed at several stages of a MDD process. In this paper, we propose the use of a Web Usability Model that can be applied in the following phases of a MDD process: **i)** in the PIM, to assess different models that specify the Web application independently of platform details (e.g., navigational models, models that represent the abstract UI); **ii)** in the PSM, to assess the concrete interface models related to a specific platform (if they exist); and **iii)** in the CM, to assess the final UI (see Fig. 1).

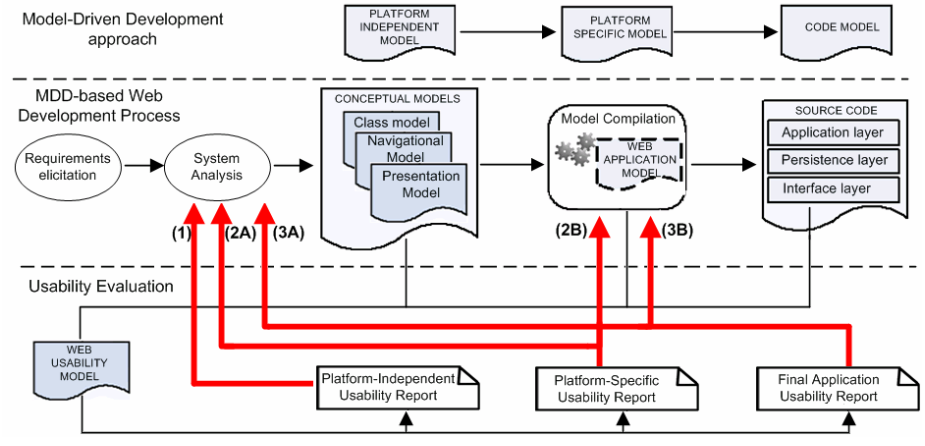

Fig. 1. Integrating a Web Usability Model into the MDD process

It should be noted that the MDD process is driven by the PIM, which is automatically transformed into a PSM, and this PSM into source code. Therefore, the evaluations performed at the PIM produce a *platform-independent usability report* that provides feedback to the system analysis stage (Fig. 1 (1)). Changes in the PIM are reflected in the CM by means of model transformations and explicit traceability between models. This prevents of usability problems to arising in the generated Web application (CM).

There are some usability attributes (e.g., degree of attractiveness) that can only be evaluated on a specific platform and taking into account the specific components of the interface (PSM) or the components that build the final UI (CM). Thus, evaluations performed at the PSM produce a *platform-specific usability report*. If the PSM does not have the required level of usability, this report will suggest changes to correct the following: the PIM models (Fig. 1 (2A)), the transformation rules that transform the PIM models into PSM models, and/or the PSM itself (Fig. 1 (2B)).

Nevertheless, the evaluations at the PIM or PSM level should be done in an iterative way until these models have the required level of usability. This allows usability evaluations at early stages in the Web development process.

Finally, evaluations performed at the CM level produce a *final application usability report*. Rather than suggest changes to improve the final UI (CM), as is usual in other approaches, this report will suggest changes to correct the PIM models (Fig. 1 (3A)), the transformation rules, and/or the PSM models (Fig. 1 (3B)).

4 Web Usability Model

The proposed Web Usability Model is an adaptation and extension of the usability model for model-driven development processes presented in Abrahão and Insfran [3]. The model was adapted to be compliant with the Software Quality Model proposed in the ISO/IEC 25000 (SQuaRE) [13]. Thus, in this section, we first introduce the ISO/IEC SQuaRE followed by a brief description of the main sub-characteristics, attributes, and Web metrics from our Web Usability Model. The entire model including all the sub-characteristics, attributes and their associated Web metrics is available at http://users.dsic.upv.es/~afernandez/WISE09/WebUsabilityModel.

4.1 ISO/IEC 25000 (SQuaRE)

The ISO/IEC 25000, also known as SQuaRE (Software Product Quality Requirements and Evaluation), was created for the purpose of providing a logically organized, enriched, and unified series of standards covering two main processes: software quality requirements specification and software quality evaluation. Both of these processes are supported by a software quality measurement process. This standard replaces the previous ISO/IEC 9126 [9] and ISO/IEC 14598 [12] standards.

In order to define our Web Usability Model, we have paid special attention to the Quality Model Division (ISO/IEC 2501n). SQuaRE proposes three different views for a quality model. These views are related to the context where the model will be applied: *Software Quality Model* to evaluate a concrete software product; *Data Quality Model* to evaluate the quality of the data managed in the product; and *Quality in Use Model* to evaluate how the stakeholders achieve their goals in a specific context of use.

Our Web Usability Model is based in the *Software Quality Model*. The main quality characteristics of the software quality model are: *functionality*, *security*, *interoperability*, *reliability*, *operability (usability)* and *efficiency*. Although the term *operability* has been proposed in SQuaRE, we use the term *usability* in this work to avoid confusions in terminology. Therefore, the goal of our Web Usability Model is to extend the software quality model proposed in SQuaRE, specifically the usability characteristic, for evaluating Web artifacts that are produced throughout a model-driven development process.

4.2 Specify Sub-characteristics and Attributes

Since SQuaRE [13] only decomposes *usability* into five high-level sub-characteristics: *learnability*, *understandability*, *operability*, *attractiveness* and *compliance*, in our approach these sub-characteristics have been decomposed into

other more detailed sub-characteristics or measurable attributes taking into account the ergonomic criteria proposed in Bastien and Scapin [4].

The first three sub-characteristics of the usability are related to user performance and can be quantified using objective measures.

Learnability refers to the attributes of a Web application that facilitate learning. This sub-characteristic includes: *help facilities* such as on-line help and user manual; *predictability*, which refers to the ease with which a user can determine the result of his or her future actions; *informative feedback* in response to user actions; and *memorability* as a measure of how quickly and accurately users can remember how to use a Web application that they have used before. Table 1 shows this decomposition.

Table 1. Decomposition of the *learnability* sub-characteristic

Sub-charac.	Attribute	Meaning
1.1 Help Facilities	1.1.1 Documentation Completeness	Help documents have all information about possible actions that can be performed by the user.
	1.1.2 Multi-user Documentation	All of the kinds of users have been described with their possible actions.
1.2 Predictability	1.2.1 Icon/Link Image Significance	Capability to predict the next action according to the images.
	1.2.2 Icon/Link Title Significance	Capability to predict the next action according to the title of the links or icons.
	1.2.3 Action Determination	Capability to predict the next action according to the user expectations.
1.3 Informative Feedback		Capability to provide information about the state of the transactions (privacy, success tasks, etc.)
1.4 Memorability	1.4.1 Time to Remember	Time needed by users to remember how to use a Web application that they have used before.
	1.4.2 Accuracy	Accuracy with which users can remember how to use a Web application that they have used before.

Understandability refers to the attributes that facilitate understanding. This sub-characteristic includes: optical *legibility* of texts and images (e.g., font size, text contrast, position of the text); *readability*, which involves aspects of information-grouping cohesiveness and density; *familiarity*, the ease with which a user recognizes the UI components and views their interaction as natural; *workload reduction*, which is related to the reduction of user cognitive effort; and finally, *user guidance*, which is related to message quality, immediate feedback, and navigability. Table 2 shows this decomposition.

Operability refers to the attributes that facilitate user control and operation. This sub-characteristic includes: *execution facilities* such as compatible browsers or plug-ins needed; *data validity* of the user inputs; *controllability* of the services execution such as cancel and undo support; *capability of adaptation* which refers to the capacity of the Web application to be adapted to the users' needs and preferences; *consistency* in the execution of services and control behavior; *error management*; and *Web application state monitoring*. Table 3 shows this decomposition.

Table 2. Decomposition of the *understandability* sub-characteristic

Sub-charac.	Attribute	Meaning
2.1 Legibility	2.1.1 Font Size	Adequacy of the font size to the context.
	2.1.2 Contrasting Text	Text always properly visible.
	2.1.3 Disposition	Position of the text in order to be visible in any situation.
2.2 Readability	2.2.1 Information-Grouping Cohesiveness	The degree to which the information is presented in groups with a thematic focus.
	2.2.2 Information Density	Amount of information needed to prevent overloads.
	2.2.3 Information Complexity	Difficulty of understanding the information provided.
2.3 Familiarity	2.3.1 Labeling Significance	Use of labels that are easily recognizable.
	2.3.2 Internationalization	Use of elements that follow standards.
	2.3.3 Metaphor	Use of metaphors to help make the interaction more natural.
2.4 Workload Reduction	2.4.1 Brevity	Reduction of cognitive effort (i.e., actions in a few steps).
	2.4.2 Self-descriptiveness	Elements are shown as concisely as possible.
2.5 User Guidance	2.5.1 Message Quality	The messages are useful for the user to interact correctly.
	2.5.2 Immediate feedback	Guides users to determine what the progress of their actions is.
	2.5.3 Navigability	Ease with which the user moves the content by accessing the Web information that is relevant.

Table 3. Decomposition of the *Operability* sub-characteristic

Sub-charac.	Attribute	Meaning
3.1 Execution facilities	3.1.1. Ease of Installation	The need to install other software components for proper operation.
	3.1.2. Multiplicity	Web application is displayed correctly on different web browsers.
	3.1.3. Updateability	The latest version is always used.
	3.1.4. Update Transparency	The user does not perform manual updates.
3.2 Data Validity		Mechanisms are provided to verify the validity of the user data input.
3.3 Controllability	3.3.1. Edition Deferral	Content inserted can be edited at any time.
	3.3.2. Cancel Support	The actions can be canceled without harmful effects to normal operation.
	3.3.3. Explicit Execution	Information about the actions being carried out is not hidden.
	3.3.4. Interruption Support	The actions can be interrupted without harmful effects to normal operation.
	3.3.5. Undo Support	The actions can be undone without harmful effects to normal operation.

Table 3. (*continued*)

Sub-charac.	Attribute	Meaning
	3.3.6. Redo Support	The actions can be redone for the user to save work.
3.4 Capability of Adaptation	3.4.1. Adaptability	Ability of the Web application to be adapted by users.
	3.4.2. Adaptivity	Ability of the Web application to suit the needs of different users.
3.5 Consistency	3.5.1. Behavior of Controls	Controls always have the same behavior.
	3.5.2. Permanence of Controls	Controls appear if their associated actions can be performed.
	3.5.3. Stability of Controls	Controls perform the actions correctly.
	3.5.4. Order Consistency	Controls are always in the same order so as not to confuse the user.
	3.5.5. Label Consistency	The labels correspond to the actions they represent.
3.6 Error Management	3.6.1. Error Prevention	Capacity to provide mechanisms to prevent common mistakes.
	3.6.2. Error Recovery	Capacity to recover from errors.
3.7 Web application State Monitoring		It is allowed to know information about running processes.

The last two sub-characteristics of the usability are related to the perception of the end-user (**attractiveness**) or evaluator (**compliance**) using the Web Application. This perception is mainly measured using subjective measures. However, some aspects of *attractiveness* that are related to aesthetic design can also be quantified by measuring the *user interface uniformity* in terms of font color, font style, font size, and the position of elements. *Compliance* can be measured by assessing the agreement of the proposed Web Usability Model with respect to the SQuaRE and other Web-design style guides.

4.3 Incorporating Web Metrics to the Usability Model

Once the sub-characteristics and attributes have been identified, Web metrics are associated to the measurable attributes in order to quantify them. The values obtained from the metrics, and the establishment of thresholds for these values, will allow us to determine the degree to which these attributes help to achieve a usable Web application. The metrics included in our model were extracted and adapted from the survey presented in Calero *et al.* [6] and other sources. The metrics selected for this work were mainly the ones that were theoretically and/or empirically validated.

Each metric was analyzed taking into account the criteria proposed in SQuaRE: such as its purpose, its interpretation, its measurement method, the measured artifact, the validity evidence, etc. If the Web metric is applied to the final source code, we also analyzed the possibility of adapting it to be applied at the PIM and/or PSM levels.

Due to space constraints, we illustrate only with some examples how we define and associate the metrics to the attributes of the Web Usability Model:

- *Color Contrast* [20] (attached to attribute 2.1.2 from Table 2): Given two colors (C_1 and C_2) with their RGB codes in decimal notation, the contrast is determined by the formula:

$$\sum | C_1(i) - C_2(i) | \text{ let } i=\{\text{Red Value, Green Value, Blue Value}\} \quad (1)$$

  (Scale type: absolute value greater or equal than 0). Applicable at PIM (if color features are defined in the abstract UI) and applicable at CM (if color features are defined in Cascading Style Sheets files). The interpretation is: the larger the value of the metric, the better color contrast.
- *Breadth of a navigational map* [1] (attached to attribute 2.5.3 from Table 2): Total number of first-level navigational targets[1] in a navigational map. (Scale type: absolute value greater than 0). Applicable at PIM. The interpretation is: the larger the value of the metric (for first-level navigational targets that represent a navigation menu), the harder it is for the user to understand the functionalities of the Web application (several options at once).
- *Depth of a navigational map* [1] (attached to attribute 2.5.3 from Table 2): The longest distance of a root navigational target to a leaf navigational target (Scale type: absolute value greater than 0). Applicable at PIM. It indicates the ease with which a navigational target can be reached (number of steps) and the likely importance of its content. The interpretation is: the larger the distance of a leaf navigational target from the root, the harder it is for the user to reach it.
- *User operation cancellability* [9] (attached to attribute 3.3.2 from Table 3): Ratio between the number of implemented functions that can be cancelled by the user prior to completion and the total number of functions requiring the pre-cancellation capability (Scale type: ratio between 0 and 1). Applicable at PIM, PSM and CM.

5 Applying the Web Usability Model

The Web Usability Model has been operationalized into a concrete Web development method based on the MDD approach. Such an operationalization means specifying the correspondences between the attributes (and their associated metrics) of the Web Usability Model and the elements of the artifacts (PIM, PSM or CM) produced by a specific model-driven Web development method. As an example, we have selected the artifacts produced by the Object-Oriented Hypermedia (OO-H) method to illustrate how the Web Usability Model can be applied for early usability evaluation.

5.1 OO-H Method

The OO-H method [8] provides designers with the semantics and notation for developing Web applications. The method includes: a design process, a pattern catalog, a *Class Diagram*, a navigational map known as *Navigation Access Diagram*

[1] The concepts of *navigational map*, *navigational links*, and *navigational targets* refer to the OO-H method modeling primitives that are introduced in Section 5.1. They mainly refer to the fact that the navigation in a Web application corresponds to a directed graph whose nodes correspond to Web pages (represented as *navigational targets* in a Navigation Model) and whose arcs correspond to *navigational links* between these pages.

(NAD), and an *Abstract Presentation Diagram* (APD). The VisualWADE tool (www.visualwade.com) automates the entire OO-H development process.

The OO-H development process starts from the domain information structure that is captured in an UML-compliant class diagram. From there, we can create different NAD instances that represent the navigation dimension. A single NAD is a partial view from the class diagram. It structures the navigational view of the Web application for a specific kind of user. The APD is based on the concept of templates and is directly derived from the NAD. The APD can be refined by applying a pattern catalog. This catalog contains a set of constructs that effectively solve problems identified within Web environments. Once the APD is refined, we can automatically generate a front-end Web application (static or dynamic) for the desired environment, such as HTML, WML, active server pages (ASPs), and JavaServer pages (JSPs).

Several concepts related to the navigational model (NAD) and the abstract UI model (APD) are introduced below to facilitate the explanation about how Web metrics can be applied to them.

Each *navigational map* (or a NAD for a specific user) has a unique *entry point* that indicates the starting point of the navigation process. A *navigational map* is made up of a set of navigational elements that can be specialized as *navigational nodes* and/or *navigational links*. A *navigational node* represents a view over the UML class diagram. A navigational node can be a *navigational target*, a *navigational class*, or a *collection*. A *navigational target* groups elements of the model (i.e., *navigational classes, navigational links and collections*) that collaborate in the coverage of a user navigational requirement. A *navigational class* represents a view over a set of attributes (*navigational attribute*) and operations (*navigational operation*) of a class from the UML class diagram. A *collection* is a hierarchical structure that groups a set of navigational links. *Navigational links* define the navigation paths that the user can follow through the UI. There are two types of links: *source links* when new information is shown at the same view (depicted as an empty arrow); and *target links* when new information is shown at another view (depicted as a bold arrow).

Each *Abstract Presentation Diagram* (APD) represents an *abstract page collection*. An *abstract page* is a set of related information that is shown at the same navigation step. A first version of an APD is automatically generated from NAD, and it can be refined by a pattern catalog.

5.2 A Case Study

As an example, we show how several metrics from the Web Usability Model can be applied to a Web application developed using the OO-H method, which is supported by the VisualWADE tool. The selected Web application is a task management system developed for a Web development company located in Alicante, Spain. The documentation used to apply the model included: the OO-H conceptual model (class, navigational, and abstract presentation diagrams) of the Task Management Web application and the Web Usability Model.

Figure 2 shows a fragment of the navigational model (NAD) that represents the access to the Web application.

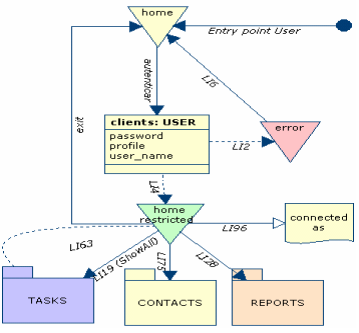

Fig. 2. First-level NAD for the Task Management Web application

Figure 2 shows the navigational access diagram (NAD) at level 0 for the *User* user. The *Entry point User* indicates the entry point to the Web application. The *Clients* navigational class corresponds to a form where the user will log into the system. This process involves a profile, a user name and a password. If the *User* exists, then a menu with a link to each one of the three identified navigational targets (packages Tasks, Contacts and Reports) will be shown (*home restricted* collection). Each navigational target can be further expanded to show detailed internal navigation. If the *User* does not exist, then an error message is shown. As an example, we can apply the following metrics:

- *Breadth of a navigational map* (see Section 4.3): the value of this metric is 4 since there are four navigational targets that the user can access (*Home*, *Tasks*, *Contacts* and *Reports*).
- *Depth of a navigational map* (see Section 4.3): the value of this metric is 2 since the longest distance of the *home* root navigational node to a leaf navigational node has 2 navigational links (*LI4* and one of the navigational links: *LI19*, *LI75* or *LI28*).

The values obtained are relevant to the Web application navigability. If the navigational map is too narrow and too deep, users will have to *click* several times and navigate through several levels to find what they are looking for, if the navigational map is too wide and shallow, users may get lost due to the excessive amount of information that can be accessed. For the type of Web application used in our case study, with a hierarchical structure, values lower than 5 levels of depth and values lower than 9 levels of breadth have been established as thresholds. Since the metrics used in this example obtains values below these thresholds, the navigational map helps to achieve a suitable level of navigability.

Figure 3 shows the PIM models (NAD and APD) that are related to the *Contacts* navigational target (a) and (b), and the final UI (CM model) (c), which is automatically generated from the APD model.

Figure 3 (a) provides detailed information about the *Contacts* navigational target. The *User* can see the information about *allContacts* or s/he can search for a given contact by providing an *initial* or by providing a *string*. These functionalities are represented by the three navigational links that connect the *contact menu* collection and the *Contact* navigational class. In addition, the user can create a new user by executing the *New* method in the *Contact1* navigational class.

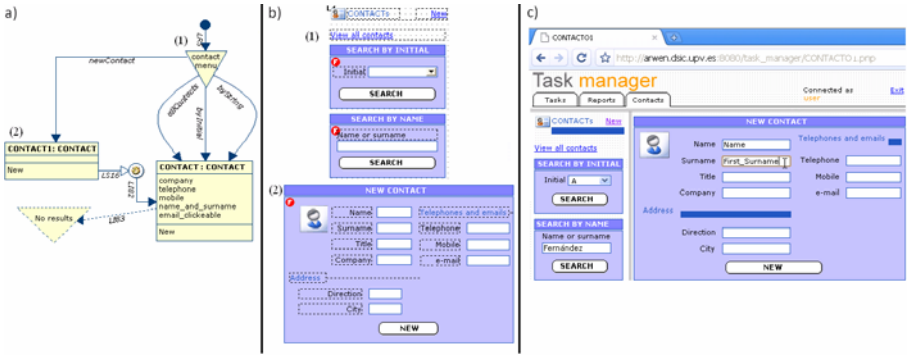

Fig. 3. (a) NAD for the Contact navigational node; (b) the APD generated from the NAD; (c) the final Web UI generated from these models (CM)

Figure 3 (b) shows a fragment of the generated APD. This model includes two abstract pages: a page for the contact menu and searching for a contact (Fig. 3 (b-1)), and a page for the execution of the *New* method from *Contact1* (Fig. 3 (b-2)). Since colors of label texts and background can be specified in this APD model (PIM), we can apply the following metric:

- *Color Contrast* (see Section 4.3): Although it should be noted that the metric must be applied to all the text elements across all the UIs, we only show, as an example, the case of the two labels *Telephones and emails* and *Address,* whose RGB color code of the text is (33, 85, 189) and the RGB color code of the background is (192, 192, 255); by applying the formula (1), the value of this metric results 332. This result is below the proposed threshold in [20] for this metric (greater or equal than 500). This color contrast can affect negatively the legibility of the text. This aspect could be improved, for instance, by changing the text color property of these labels in the APD model in order to get a better color contrast.

Finally, Figure 3 (c) shows the generated final UI. As an example, we can apply the following metric:

- *User operation cancellability* (see Section 4.3): The UI does not provide any explicit control (link or button) to allow users to cancel the insertion of a contact if they decide not to insert the contact; therefore, the value of this metric is 0/1=0. This result is negative because values that are closer to 0 indicate that the user cannot cancel transactions, which negatively affects the *controllability* sub-characteristic. This aspect could be improved, for instance, by including a navigational link in the corresponding NAD that allows to return to the *contact menu*. After correcting the NAD model, a more usable UI can be generated including the corresponding cancel link or button.

Although only an excerpt of the case study is shown, the Web Usability Model can be applied to the entire Web application. In this example, we have selected the OO-H method to operationalize the Web Usability Model; however, it is important to note that the model can be operationalized to other methodologies that follow a Web development process based on a MDD approach (e.g., UWE [14], WebML [7]).

6 Conclusions and Further Work

This paper has presented a Web Usability Model that is aligned with the ISO/IEC 25000 (SQuaRE) for evaluating and improving the usability of Web applications developed using a MDD approach. The aim of this proposed model is to perform usability evaluations not only when the Web application is completed (evaluating a CM), but also at earlier stages of the Web development (evaluating the PIM and PSM models) to provide feedback during the analysis and design stages. In this way, usability is taken into account throughout the entire process, enabling Web applications to be developed with better quality, thereby reducing effort at the maintenance stage.

The Web Usability Model can be very useful not only to evaluate PIMs, but also to discover deficiencies and/or limitations of the PIM expressiveness and the transformation rules to support some usability attributes. For instance, the OO-H method can be extended to support some usability attributes, such as Information-Grouping Cohesiveness (see Table 2, attribute 2.2.1), by incorporating mechanisms to semantically group attributes in the navigational model, thus avoiding the generation of Web forms that have a long list of attributes that are not grouped by any criteria.

We believe that the inherent features of model-driven development processes (e.g., traceability between models by means of model transformations) provide a suitable environment for performing usability evaluations. Specifically, if the usability of an automatically generated UI (e.g., using the VisualWade tool) can be assessed, the usability of any future UI produced by this tool could be predicted. In other words, we are talking about a UI that is *usable by construction* [2], at least to some extent.

Although the model has been operationalized to the OO-H method, it can also be applied to other methods such as UWE or WebML by specifying the relationships between the attributes (and their associated metrics) from the Web Usability Model with the elements of the PIM, PSM, and CM models from other methods.

Further work is intended to define a usability evaluation process including detailed guidelines on how evaluators could apply the Web Usability Model; exploring proper thresholds and aggregation mechanisms for values obtained by individual metrics; and performing analyses of the impact on how the attributes affect (negatively or positively) other attributes of the Usability Model. In addition, we plan to conduct empirical studies to confirm the relationship between the measures obtained at the PIM/PSM levels with the measures obtained when users actually use the Web application. Another aim is to develop a tool to manage the Web Usability Model creating a repository of catalogued metrics following SQuaRE patterns with the capability to support evaluations of the PIM and PSM models, as well as the final Web applications (CM).

Acknowledgments. This work is financed by META project (ref. TIN2006-15175-C05-05), the Quality-driven Model Transformation Project from the *Universidad Politécnica de Valencia*. The authors thank Jaime Gomez from *Universidad de Alicante* for his valuable help in providing the models and the generated Web application used in the case study.

References

1. Abrahão, S., Condori-Fernández, N., Olsina, L., Pastor, O.: Defining and Validating Metrics for Navigational Models. In: Proc. of the 9th Inter. IEEE Software Metrics Symposium, pp. 200–210 (2003)
2. Abrahão, S., Iborra, E., Vanderdonckt, J.: Usability Evaluation of User Interfaces Generated with a Model-Driven Architecture Tool. In: Maturing Usability. Springer HCI series, vol. 10, pp. 3–32 (2007)
3. Abrahão, S., Insfran, E.: Early Usability Evaluation in Model-Driven Architecture Environments. In: Proc. of the 6th IEEE International Conference on Quality Software, pp. 287–294. IEEE Computer Society, Los Alamitos (2006)
4. Bastien, J.M., Scapin, D.L.: Ergonomic Criteria for the Evaluation of Human-Computer Interfaces. Tech. Rep. n.156. INRIA, Rocquencourt, France (1993)
5. Becker, S.A., Mottay, F.E.: A Global Perspective on Web Site Usability. IEEE Software 18(1), 54–61 (2001)
6. Calero, C., Ruiz, J., Piattini, M.: Classifying Web Metrics Using the Web Quality Model 29(3), 227–248 (2005)
7. 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)
8. Gomez, J., Cachero, C., Pastor, O.: Conceptual Modeling of Device-Independent Web Applications. IEEE MultiMedia 8(2), 26–39 (2001)
9. ISO/IEC 9126, Software Engineering, Product Quality (2001)
10. ISO/IEC 9241, Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) (1998)
11. ISO/IEC 12207, Information Technology, Software Life Cycle Processes (1995)
12. ISO/IEC 14598, Information technology, Software Product Evaluation (1999)
13. ISO/IEC 25000, Software Product Quality Requirements and Evaluation (SQuaRE) (2005)
14. Kraus, A., Knapp, A., Koch, N.: Model-Driven Generation of Web Applications in UWE. In: 3rd Inter. Workshop on Model-Driven Web Engineering (2006)
15. Moraga, M.A., Calero, C., Piattini, M., Diaz, O.: Improving a Portlet Usability Model. Software Quality Control 15(2), 155–177 (2007)
16. Olsina, L., Rossi, G.: Measuring Web Application Quality with WebQEM. IEEE Multimedia 9(4), 20–29 (2002)
17. Seffah, A., Donyaee, M., Kline, R.B., Padda, H.K.: Usability Measurement and Metrics: A Consolidated Model. Software Quality Journal 14(2), 159–178 (2006)
18. Signore, O.: A Comprehensive Model for Web Site Quality. In: Proc. of the 7th IEEE Inter. Symposium on Web Site Evolution, pp. 30–36. IEEE Computer Society, Los Alamitos (2005)
19. Sutcliffe, A.: Assessing the Reliability of Heuristic Evaluation for Web Site Attractiveness and Usability. In: Proc. of the 35th Annual Hawaii Inter. Conf. on System Sciences, pp. 1838–1847 (2002)
20. W3C: Techniques For Accessibility Evaluation And Repair Tools. Working Draft (2000)