36

Designing Highly Usable Web Applications

Silvia Abrahão
Universitat Politècnica de València

Emilio Insfran
Universitat Politècnica de València

Adrian Fernandez
Universitat Politècnica de València

36.1 Introduction

Usability is considered to be one of the most important quality factors for Web applications, along with others such as reliability and security (Offutt 2002). 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 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.

Today's Web applications deliver a complex amount of functionality to a large number of diverse groups of users. The nature of Web development forces designers to focus primarily on time-to-market issues in order to achieve the required short cycle times. In this context, model-driven engineering (MDE) approaches seem to be very promising, since Web development can be viewed as a process of transforming a model into another model until it can be executed in a development environment. MDE has been successfully used to develop Web-based interactive applications. Existing approaches usually comprise high-level models (e.g., a presentation model, task model) that represent interactive tasks that are performed by the users of the Web application in a way that is independent from platforms and interaction modalities. From these models, the final Web application can be generated by applying model transformations.

Model-driven engineering approaches can play a key role in helping Web designers to produce highly usable Web applications. The intermediate models that comprise the Web application can be inspected

and corrected early in the development process, ensuring that a product with the required level of usability is generated. Although a number of model-driven Web development processes have emerged, there is a need for usability evaluation methods that can be properly integrated into these types of processes (Fernández et al. 2011). This chapter addresses this issue by showing how usability evaluations are integrated into a specific model-driven development method to ensure the usability of the Web applications developed.

36.2 Underlying Principles

Traditional Web development approaches do not take full advantage of the usability evaluation of interactive systems based on the design artifacts that are produced during the early stages of the development. These intermediate software artifacts (e.g., models and sketches) are usually used to guide Web developers in the design but not to perform usability evaluations. In addition, since the traceability between these artifacts and the final Web application is not well-defined, performing usability evaluations by considering these artifacts as inputs may not ensure the usability of the final Web application.

This problem may be alleviated by using a MDE approach due to its intrinsic traceability mechanisms that are established by the transformation processes. The most well-known approach to MDE is the model-driven architecture (MDA) defined by the object management group—OMG (2003). In MDA, platform-independent models (PIMs) such as task models or navigation models may be transformed into platform-specific models (PSMs) that contain specific implementation details from the underlying technology platform. These PSMs may then be used to generate the source code for the Web application (Code Model—CM), thus preserving the traceability among PIMs, PSMs, and source code.

A model-driven development approach therefore provides a suitable context for performing early usability evaluations. Platform-independent (or platform-specific) models can be inspected during the early stages of Web development in order to identify and correct some of the usability problems prior to the generation of the source code. We are aware that not all usability problems can be detected based on the evaluation of models since they are limited by their own expressiveness and, most importantly, they may not predict the user behavior or preferences. However, studies such as that of Hwang and Salvendy (2010) claim that usability inspections, applying well-known usability principles on software artifacts, may be capable of finding around 80% of usability problems. In addition, as suggested by previous studies (Andre et al. 2003), the use of inspection methods for detecting usability problems in product design (PIMs or PSMs in this context) can be complemented with other evaluation methods performed with end users before releasing a Web application to the public.

36.3 State of the Art in Usability Evaluation for Web Applications

Usability evaluation methods can be mainly classified into two groups: empirical methods and inspection methods. Empirical methods are based on observing, 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 (e.g., mock-ups, conceptual models, user interfaces [UIs]), which are commonly UIs, with regard to their conformance with a set of guidelines.

The employment of usability evaluation methods to evaluate Web artifacts was investigated through a systematic mapping study in a previous work (Fernandez et al. 2011). This study revealed various findings:

1. There is lack of usability evaluation methods that can be properly integrated into the early stages of Web development processes.
2. There is a shortage of usability evaluation methods that has been empirically validated.

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 (Cockton et al. 2003). There are several proposals based on inspection methods to deal with Web usability issues, such as the cognitive walkthrough for the Web—CWW (Blackmon 2002) and the web design perspectives—WDP (Conte et al. 2007). Cognitive walkthrough for the Web assesses the ease with which a user can explore a website by using semantic algorithms. This method extends and adapts the heuristics proposed by Nielsen and Molich (1990) with the aim of drawing closer to the dimensions that characterize a Web application: content, structure, navigation, and presentation. However, these kinds of methods tend to present a considerable degree of subjectivity in usability evaluations. In addition, this method only supports ease of navigation.

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 (Ivory and Hearst 2002) and the Web quality evaluation method—WebQEM (Olsina and Rossi 2002). The WebTango methodology allows us to obtain quantitative measures, which are based on empirically validated metrics for UIs, to build predictive models in order to evaluate other UIs. Web quality evaluation method performs a quantitative evaluation of the usability characteristics proposed in the ISO 9126-1 (2001) standard, and these quantitative measures are aggregated in order to provide usability indicators.

The aforementioned inspection methods are oriented toward application in the traditional Web development context; they are therefore principally employed in the later stages of Web development processes. As mentioned earlier, 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 and Schmidt (2005), Abrahão and Insfran (2006), and Molina and Toval (2009).

Atterer and Schmidt (2005) proposed a model-based usability validator prototype with which to analyze models that represent enriched Web UIs. 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 (2006) 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 subcharacteristics proposed in the ISO 9126 standard, it is not specific to Web applications and does not provide specific metrics. The same model was used by Panach et al. (2008) with the aim of providing metrics for a set of attributes that would be applicable to the conceptual models that are obtained as a result of a specific model-driven Web development process. Molina and Toval (2009) 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. In this chapter, we address this issue by showing how usability evaluations can be integrated into a specific model-driven development method.

36.4 Early Usability Evaluation in Model-Driven Web Development

Currently, there are several Web development methods (e.g., WebML [Ceri et al. 2000], OO-H [Gomez et al. 2011]), environments (e.g., Teresa [Mori et al. 2004]), and languages (e.g., UsiXML [Limbourg et al. 2004]) that exploit MDE techniques to develop Web applications.

These approaches usually comprise high-level models that represent tasks that are performed by the Web application users in a way that is independent from the technology used.

The usability of a software product (e.g., a Web application) obtained as a result of a transformation process can be assessed at several stages of a model-driven Web development process. In this chapter, we suggest the use of a **Web Usability Model** that contains a set of usability attributes and metrics that

can be applied by the Web designer in the following artifacts obtained in a MDA-based Web development process:

- PIM, to inspect the different models that specify the Web application independently of platform details (e.g., screen flow diagrams, screen mock-ups, screen navigation diagrams)
- PSM, to inspect the concrete design models related to a specific platform, and
- CM, to inspect the UI generated or the Web application source code (see Figure 36.1)

In order to perform usability evaluations, the Web designer should select the set of relevant usability attributes and metrics from the Web Usability Model according to the purpose of the evaluation, the development phase, and the type of artifacts (models) to be inspected. There are some usability attributes that can be evaluated independently of the platform (at the PIM level), other attributes that can only be evaluated on a specific platform and taking into account the specific components of the Web application interface (at the PSM level), and finally some attributes that can be evaluated only in the final Web application (CM).

The usability evaluations performed at the PIM level produce a *platform-independent usability report* that provides a list of usability problems with recommendations to improve PIM (Figure 36.1—1A). Changes in PIM are reflected in CM by means of model transformations and explicit traceability between models (PIM to PSM and PSM to CM). This prevents usability problems from appearing in the Web application generated.

Evaluations performed at PSM produce a *platform-specific usability report*. If PSM does not allow a Web application with the required level of usability to be obtained, this report will suggest changes to correct the following: the PIM (Figure 36.1—2A), the transformation rules that transform PIM into PSM (Figure 36.1—2B), and/or the PSM itself (Figure 36.1—3B). Nevertheless, the evaluations at the PIM or PSM level should be done in an iterative way until these models allow the generation of a Web application with the required level of usability. This allows usability evaluations to be performed at early stages of a Web development process.

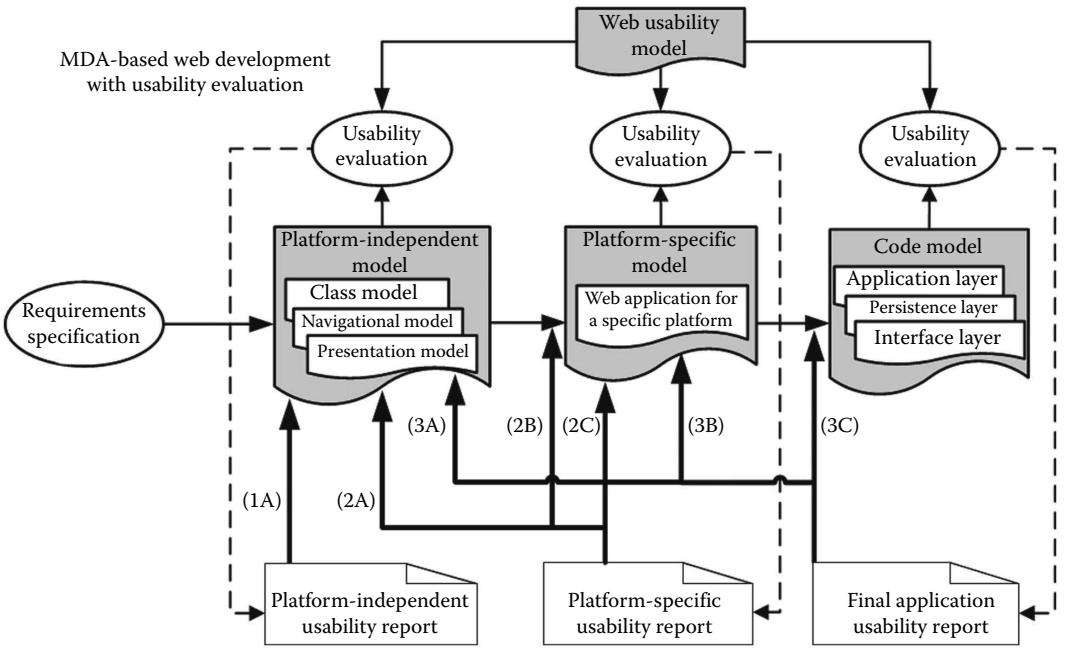

FIGURE 36.1 Integrating usability evaluations in Model-driven Web development.

Finally, evaluations performed at the CM level produce a *final application usability report*. Rather than suggesting changes to improve the final Web application, as is usual in other approaches, this usability report suggests changes to correct the PIM (Figure 36.1—3A), the PSM (Figure 36.1—3B), and/or the transformation rules that transforms the PSM into the final Web application (Figure 36.1—3C).

Our evaluation strategy is aimed at providing support to the intrinsic usability of the Web application generated by following a model-driven development process, and to the notion of usability proven by construction (Abrahão et al. 2007). Usability by construction is analogous to the concept of correctness by construction (Hall and Chapman 2002) introduced to guarantee the quality of a safety-critical system. In this development method, the authors argue that to obtain software with almost no defects (0.04% per thousand lines of code; KLOC), each step in the development method should be assessed with respect to correctness. If we can maintain proof of the correctness of a software application from its inception until its delivery, it would mean that we can prove that it is correct by construction. Similarly, if we can maintain proof of the usability of a Web application from its model specification until the source code generation, it would mean that we can prove it is usable by construction. Of course, we can only hypothesize that each model may allow a certain level of usability in the generated Web application to be reached. Therefore, we may predict the global usability of an entire Web application by estimating the relative usability levels that the models and transformations involved in a specific model-driven Web development method allow us to accomplish. We cannot prove that a Web application is totally usable (i.e., it satisfies all the usability attributes for all the users), but we can prove that it is usable at a certain level.

In the following section, we provide a brief overview of our Web usability model and show how it can be used to evaluate the usability of a Web application, early in the Web development process.

36.5 Web Usability Model

The objective of the Web usability model is to help Web designers and developers to achieve the required Web application usability level through the definition of usability characteristics and attributes, measurement of usability attributes, and evaluation of the resulting usability.

The Web usability model (Fernandez et al. 2009) is an adaptation and extension of the usability model for model-driven development processes, proposed by Abrahão and Insfran (2006). The model was adapted to be compliant with the ISO/IEC 25010 (2011) standard, also known as SQuaRE (software product quality requirements and evaluation). This standard 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. SQuaRE replaces the previous ISO/IEC 9126 (2001) and ISO/IEC 14598 (1999) standards.

In order to define the Web usability model, we have paid special attention to the SQuaRE quality model division (ISO/IEC 2501n), where three different quality models are proposed: the software product quality model, the system quality in use model, and the data quality model. Together, these models provide a comprehensive set of quality characteristics relevant to a wide range of stakeholders (e.g., software developers, system integrators, contractors, and end users). In particular, the software quality model defines a set of characteristics for specifying or evaluating software product quality; the data quality model defines characteristics for specifying or evaluating the quality of data managed in software products; and the quality in use model defines characteristics for specifying or evaluating the quality of software products in a particular context of use.

The goal of the Web usability model is to extend the software quality model proposed in SQuaRE, specifically the usability characteristic, for specifying, measuring, and evaluating the usability of Web applications that are produced throughout a model-driven development process from the end user's perspective.

In the following, we introduce a brief description of the main subcharacteristics, usability attributes, and metrics of our Web usability model. The entire model, including all the subcharacteristics, usability attributes, and their associated metrics is available at http://www.dsic.upv.es/~afernandez/WebUsabilityModel.

36.5.1 Usability Attributes

SQuaRE decomposes usability into seven high-level subcharacteristics: appropriateness recognizability, learnability, operability, user error protection, accessibility, UI aesthetics, and compliance. However, these subcharacteristics are generic and need to be further broken down into measurable usability attributes. For this reason, our proposed Web usability model breaks down these subcharacteristics into other subcharacteristics and usability attributes in order to cover a set of Web usability aspects as broadly as possible. This breakdown has been done by considering the ergonomic criteria proposed by Bastien and Scapin (1993) and other usability guidelines for Web development (Lynch and Horton 2002; Leavit and Shneiderman 2006).

The first five subcharacteristics are related to user performance and can be quantified using objective metrics.

Appropriateness recognizability refers to the degree to which users can recognize whether a Web application is appropriate for their needs. In our Web usability model, this subcharacteristic was broken down by differentiating between those attributes that enable the optical *legibility* of texts and images (e.g., font size, text contrast, position of the text), and those attributes that allow information *readability*, which involves aspects of information grouping cohesiveness, information density, and pagination support. In addition, it also includes other subcharacteristics such as *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; *user guidance*, which is related to message availability and informative feedback in response to user actions; and *navigability*, which is related to how easily the content is accessed by the user.

Learnability refers to the degree to which a Web application facilitates learning about its employment. In our model, this subcharacteristic was broken down into other subcharacteristics such as *predictability*, which refers to the ease with which a user can determine the result of his/her future actions; *affordance*, which refers to how users can discover which actions can be performed in the next interaction steps; and *helpfulness*, which refers to the degree to which the Web application provides help when users need assistance.

Several of the aforementioned concepts were adapted from the affordance term that has been employed in the Human–Computer Interaction field in order to determine how intuitive the interaction is (Norman 1988). These subcharacteristics are of particular interest in Web applications. Users should not spend too much time learning about the Web application employment. If they feel frustrated when performing their tasks, it is likely that they may start finding other alternatives.

Operability refers to the degree to which a Web application has attributes that make it easy to operate and control. In our model, this subcharacteristic was broken down into other subcharacteristics related to the technical aspects of Web applications such as *compatibility* with other software products or external agents that may influence the proper operation of the Web application; *data management* according to the *validity of input data* and its *privacy*; *controllability* of the action execution such as cancel and undo support; *capability of adaptation* by distinguishing between *adaptability*, which is the Web application's capacity to be adapted by the user; and *adaptivity*, which is the Web application's capacity to adapt to the users' needs (i.e., the difference is in the agent of the adaptation); and *consistency* in the behavior of links and controls.

User error protection refers to the degree to which a Web application protects users against making errors. In the ISO/IEC 9126-1 (2001) standard, this subcharacteristic was implicit in the operability term. However, the ISO/IEC 25010 (SQuaRE) standard made it explicit since it is particularly important to achieve freedom from risk. In our model, this subcharacteristic was broken down into other subcharacteristics related to *error prevention* and *error recovery*.

Accessibility refers to the degree to which a Web application can be used by users with the widest range of characteristics and capabilities. Although the concept of accessibility is so broad that it may

require another specific model, the SQuaRE standard added this new subcharacteristic as an attempt to integrate usability and accessibility issues. In our model, this subcharacteristic was broken down into usability attributes by considering not only a range of human disabilities (e.g., blindness, deafness), but also temporary technical disabilities (e.g., element unavailability, device dependency). The usability attributes include *magnifier support*, which indicates that the text of a Web page must be resized regardless of the options offered by the browser for this action; *device independency*, which indicates that the content should be accessible regardless of the type of input device employed (mouse, keyboard, voice input); and *alternative text support*, which indicates that the multimedia content (images, sounds, animations) must have an alternative description to support screen readers and temporary unavailability of these elements.

The last two usability subcharacteristics are related to the perception of the end user (UI aesthetics) or evaluator (compliance) using the Web application. This perception is mainly measured using subjective metrics.

User interface aesthetics refers to the degree to which UI enables pleasing and satisfying interaction for the user. This definition evolved from the attractiveness characteristic proposed in the ISO/IEC 9126 (2001) standard. Although this subcharacteristic is clearly subjective and can be influenced by many factors in a specific context of use, it is possible to define attributes that might have a high impact on how users perceive the Web application.

In our model, this subcharacteristic was broken down into other subcharacteristics related to the *uniformity* of the elements presented in UI (e.g., font, color, position); *interface appearance customizability*, which should not be confused with the subcharacteristic *capability of adaption*, since it is not related to user needs, but to aesthetic preferences; and *degree of interactivity*, whose definition was proposed by Steuer (1992) as "the extent to which users can participate in modifying the form and content of a media environment in real time." This is a concept that has recently become increasingly important owing to collaborative environments and social networks through the Web.

Compliance refers to how consistent the Web application is with regard to rules, standards, conventions, and design guidelines employed in the Web domain. In our model, this subcharacteristic was broken down in other subcharacteristics such as the degree of fulfillment to the ISO/IEC 25010 (2011) standard and other relevant usability and Web design guidelines.

36.5.2 Usability Metrics

Once the subcharacteristics and usability attributes have been identified, metrics are associated to the measurable attributes in order to quantify them. The values obtained from the metrics (i.e., measures), 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. A measure is therefore a variable to which a value is assigned as the result of measurement (ISO/IEC 9126 2001).

The metrics included in this usability model were extracted and adapted from several sources: a survey presented by Calero et al. (2005), the ISO/IEC 25010 standard (2011), and the Web design and accessibility guidelines proposed by the W3C Consortium (2008). The metrics selected for our model were mainly the ones that were theoretically and/or empirically validated. Each metric was analyzed taking into account the criteria proposed in SQuaRE (e.g., its purpose, its interpretation, its measurement method, the measured artifact, the validity evidence). If the Web metric was applied to the final UI (CM), we also analyzed the possibility of adapting it to the PIM and/or PSM levels. Due to space constraints, we have only illustrated some examples of how we defined and associated metrics to the attributes of the Web usability model. Table 36.1 shows some examples of metrics for some selected usability attributes. It also shows at what level of abstraction the metric can be applied.

TABLE 36.1 Examples of Usability Metrics

Subcharacteristic/ Usability Attribute	Metric	Definition	Model
Learnability/ Predictability/Icon/ Link Title Significance	Proportion of titles chosen suitably for each icon/title	Ratio between the total number of titles that are appropriate to the link icon they are associated and the total number of titles associated with existing links or icons (scale: $0 < X \leq 1$)	PIM, PSM, or CM
Appropriateness recognizability/User guidance/Quality of messages	Proportion of meaningful messages (error, advise, and warning messages)	Ratio of error messages explaining an error in a proper way and the total contexts where this error may occur (scale: $0 < X \leq 1$)	PIM, PSM, or CM
Appropriateness recognizability/ Navigability/ Reachability	Breadth of the internavigation (BiN)	Level of breadth in the user navigation. It represents the different paths that can be selected by the user in a certain context of the user navigation (i.e., homepage, internal sections) (scale: Integer > 0)	PIM
	Depth of the navigation (DN)	Level of depth in the user navigation. It indicates the longest navigation path (without loops), which is needed to reach any content or feature (scale: Integer > 0)	PIM

36.6 Operationalizing the Web Usability Model

The operationalization of the Web usability model consists of the instantiation of the model to be used in a specific model-driven Web development method. This is done by defining specific measurement functions (calculation formulas) for the generic metrics of the usability model, taking into account the modeling primitives of the PIMs and PSMs of the Web development method.

36.6.1 Selecting the Web Development Process

In this chapter, we use the Object-Oriented Hypermedia method—OO-H (Gómez et al. 2001) as an example of operationalization of the Web usability model. OO-H provides the semantics and notation for developing Web applications. The PIMs that represent the different concerns of a Web application are the class model, the navigational model, and the 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 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.

36.6.2 Applying the Web Usability Model in Practice

We applied the Web usability model (operationalized for the OO-H method) to a Task Manager Web application designed for controlling and monitoring the development of software projects in a software company.

A Web development project involves a series of *tasks* that are assigned to a *user*, who is a programmer in the company. For each task, the start date, the estimated end date, priority, etc. are recorded. The project manager organizes the tasks in folders according to certain criteria: pending tasks, critical tasks, and so on. Additionally, it is possible to attach external *files* to a task (e.g., requirements documents, models, code). Programmers can also add *comments* to the tasks and send *messages* to other programmers. Every day, programmers can generate a report (*Daily Report*), including information related to the tasks they are working on. Finally, the customers (i.e., stakeholders) of the project are recorded as *Contacts* in the Web application.

Due to space constraints, we focus only on an excerpt of the Task Manager Web application: the user access to the application, the task management, and the registration of customer's (contact's) functionalities. Figure 36.2a shows a fragment of the NAD that represents the user access to the application, whereas Figure 36.2b shows the abstract presentation model generated from the class model and the navigational model.

The NAD has a unique *entry point* that indicates the starting point of the navigation process. This diagram is composed of a set of navigational elements that can be specialized as navigational nodes or navigational links. A *navigational node* represents a view in 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 attributes) and operations (navigational operations) 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 UI. There are two types of links: *source links* when new information is shown in the same view (depicted as an empty arrow); and *target links* when new information is shown in another view (depicted as a filled arrow).

In Figure 36.2a, NAD shows the programmer (*User*) accessing the Web application, starting at the *User Entry Point*. The next node in the diagram is the *User* navigational class, which corresponds to a data entry form through which the programmer can log into the application. This process requires *profile* information, a *username*, and a *password*. If the user exists, which is represented by the navigational link *LI4*, the system displays a menu (*restricted home*) with links to each of the three possible navigational destinations (*Tasks*, *Contacts*, and *Reports*). If the user does not exist, the system shows an error message (*LI2*) returning to the initial data entry form (LI6). The *restricted home* menu is represented by a collection node, which includes the navigational links LI63, LI19, LI75, and LI28. In addition, the label *connected as* shows the log in information (LI92).

In Figure 36.2b, the abstract presentation model shows the three abstract pages generated from the previous NAD. The *User* navigational class produces a Web page showing the abstract access to the application (see Figure 36.2—b1). The navigational link *LI2* generates the error message represented by the *error* collection in a different abstract Web page (see Figure 36.2—b2). Finally, the *restricted home* collection, which is accessed through the navigational link *LI4*, generates the user's home page in a different abstract Web page (see Figure 36.2—b3), representing the possible navigational targets (*Tasks*, *Reports*, and *Contacts*). However, as the label *connected as,* represented by the navigational link *LI96*, is a *source link* (empty arrow), the log in information is shown in the same user's generated home page (see Figure 36.2—b3).

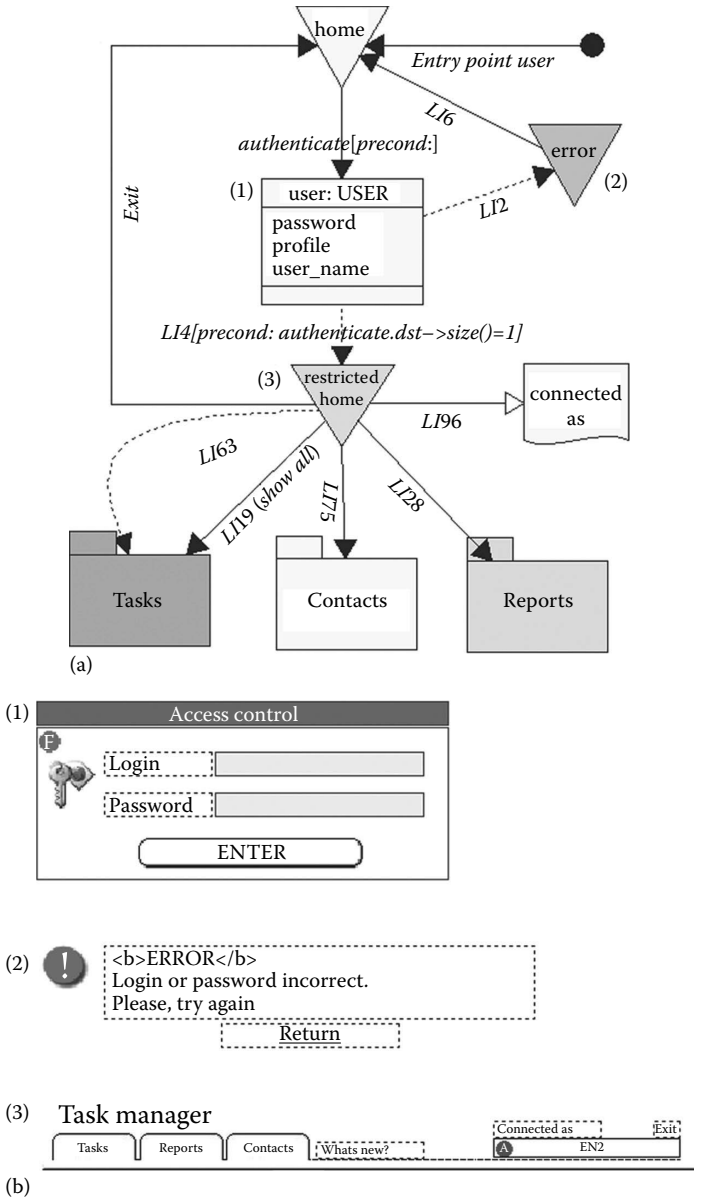

FIGURE 36.2 (a) First level NAD and (b) its associated APD for the Task Manager Web application.

The following usability metrics can be applied to the Navigational Model in Figure 36.2a:

- *Breadth of the internavigation (see Table 36.1)*: This metric can only be operationalized in those NADs that represent the first navigation level (i.e., internavigation). The calculation formula to be applied is *BiN (NAD) = Number of output target links from collections connected to navigational targets*. The thresholds* defined for the metric are [BiN = 0]: critical usability problem; [1 ≤ BiN ≤ 9]: no usability problem; [10 ≤ BiN ≤ 14]: low usability problem; and [15 ≤ BiN ≤ 19]: medium

* These thresholds were established considering hypertext research works such as those of Botafogo et al. (1992), and usability guidelines such as those of Leavit and Shneiderman (2006) and Lynch and Horton (2002).

usability problem. Therefore, the breadth of the internavigation for NAD shown in Figure 36.2a is 4 since there are four first-level navigational targets (i.e., Home, Tasks, Contacts, and Reports). This signifies that no usability problem was detected.

- *Depth of the navigation (see Table 36.1)*: This metric considers the number of navigation steps from the longest navigation path. Therefore, the formula to be applied is *DN (NAD) = Number of navigation steps from the longest navigation path,* where "navigation step" is when a target link exists between two nodes and "longest navigation path" is the path with the greatest number of navigation steps, which begins in the first navigational class or collection where navigation starts, and which ends in the last navigational class or service link, from which it is not possible to reach another previously visited modeling primitive. The thresholds* defined for this metric are [1 ≤ DN ≤ 4]: no usability problem; [5 ≤ DN ≤ 7]: low usability problem; [8 ≤ DN ≤ 10]: medium usability problem; and [DN ≥ 10]: critical usability problem. Therefore, the depth of the navigation for the NAD shown in Figure 36.2a is 2, because the length of the longest navigational path between the root of the navigational map and its leaves (Tasks, Contacts, and Reports—navigational targets) passes through two navigational links. This means that no usability problem was detected since obtained values are within the threshold [1 ≤ BaN ≤ 9].

The values obtained through these metrics are of great relevance to the navigability of a Web application. If the navigational map of the Web application is too narrow and deep, users may have to click too many times and navigate through several levels to find what they are looking for. However, if the navigational map is too wide and shallow, users may be overwhelmed due to the excessive amount of information they can access. In this context, for large Web applications that have a hierarchical structure, the recommended value is usually a *depth* of less than 5 levels and an *amplitude* of less than 9 levels. Since the navigational map of our Task Manager Web application obtains values below these thresholds, this Web application has a good navigability.

Regarding the abstract presentation model in Figure 36.2b, the following metrics, which are closer to the final UI, may be applied:

- *Proportion of meaningful messages (see Table 36.1)*: There is only one error message "Log in or password incorrect. Please try again." As the message properly explains what is wrong, the value of the metric is 1/1 = 1. This result is very positive because values closer to 1 indicate that the error messages are very significant, thus guiding the user toward the correct use of the Web application.
- *Proportion of titles chosen suitably for each icon/title (see Table 36.1)*: As an example, we focus on the titles of the links of the APD shown in Figure 36.2b: *Enter, Return, What's new,* and *Exit*. Since all the titles make us intuitively predict the target of these links, the value for this metric is 4/4 = 1. This result is very positive because values closer to 1 indicate that the selected titles are appropriate to the icons or links, allowing the user to predict what actions will take place.

Obviously, these metrics can be applied to each one of the abstract Web pages of an application to determine to which extent they provide proper labels and meaningful error messages. In this running example, we have used only a fragment of the Task Manager Web application. However, the Web usability model is fully applicable to a complete Web application that has been developed following a MDE approach. Although the usability model has been operationalized in OO-H, it can also be used with other model-driven Web development methods, for example, WebML (Ceri et al. 2000) or UWE (Kraus et al. 2006). The key is to relate the attributes of the Web usability model with the modeling primitives of the PIMs, PSMs, or the final code of the Web application. To establish these relationships, it is essential to understand the purpose of the generic usability metrics (to understand what concepts they are intended to measure) and then to identify which elements of the models provide the semantics needed to support them.

* These thresholds were established considering hypertext research works such as those of Botafogo et al. (1992), and usability guidelines such as those of Leavit and Shneiderman (2006) and Lynch and Horton (2002).

36.7 Summary

This chapter introduced a usability inspection strategy that can be integrated into specific model-driven Web development methods to produce highly usable Web applications. This strategy relies on a usability model that has been developed specifically for the Web domain and which is aligned with the SQuaRE standard to allow the iterative evaluation and improvement of the usability of Web-based applications at the design level. Therefore, this strategy does not only allow us to perform usability evaluations when the Web application is completed, but also during the early stages of its development in order to provide feedback to the analysis and design stages.

The inherent features of model-driven development (e.g., well-defined models and the traceability between models and source code established by model transformations) provide a suitable context in which to perform usability evaluations. This is due to the fact that the usability problems that may appear in the final Web application can be detected early and corrected at the model level. The model-driven development also allows the automation of common usability evaluation tasks that have traditionally been performed by hand (e.g., generating usability reports that include recommendations for improvement).

From a practical point of view, our usability inspection strategy enables the development of more usable Web-based applications by construction, meaning that each step of the Web development method (PIM, PSM, Code) satisfies a certain level of usability, thereby reducing the effort of fixing usability problems during the maintenance phase. This is in line with research efforts in the computer science community toward raising the level of abstraction not only of the artifacts used in the development process but also in the development process itself.

In this context, there are several open research challenges, mainly due to the highly evolving nature of the Internet and Web applications, and also because of the concept of usability. Although there are some underlying usability concepts that may remain immutable (e.g., the concept of navigation), there are new devices, platforms, services, domains, technologies, and user expectations that must be taken into consideration when performing usability inspections. This may lead us to determine the families of Web applications and to identify the most relevant usability attributes for these families. In addition, there is a need for the study 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.

36.8 Future Research Opportunities

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 those of Juristo et al. (2007) 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 review that we performed to investigate which usability evaluation methods have been used to evaluate Web artifacts and how they were employed (Fernández et al. 2011). 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 be properly integrated at relevant points of a Web development process.

Glossary

Model-driven engineering (MDE): A software development approach that focuses on creating and exploiting domain models (abstract representations of the knowledge and activities that govern a particular application domain), rather than on the computing (or algorithmic) concepts.

Platform-independent model (PIM): A view of a system that focuses on the operation of a system while hiding the details necessary for a particular platform.

Platform-specific model (PSM): A view of a system from the platform-specific point. A PSM combines the specifications in the PIM with the details that specify how that system uses a particular type of platform.

Usability by construction: Proof of the usability of a software application from its model specification to its generated final code.

Usability problem: The problem that users will experience in their own environment, which affects their progress toward goals and their satisfaction.

Acknowledgments

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

References

Abrahão, S. and Insfran, E. 2006. Early usability evaluation in model-driven architecture environments. In *6th IEEE International Conference on Quality Software (QSIC'06)*, Beijing, China, pp. 287–294.

Abrahão, S., Iborra, E., and Vanderdonckt, J. 2007. Usability evaluation of user interfaces generated with a model-driven architecture tool. In Law, E. L-C., Hvannberg, E., and Cockton, G. (Eds.), *Maturing Usability: Quality in Software, Interaction and Value*, Springer, New York, pp. 3–32.

Andre, T.S., Hartson, H.R., and Williges, R.C. 2003. Determining the effectiveness of the usability problem inspector: A theory-based model and tool for finding usability problems. *Human Factors* 45(3): 455–482.

Atterer, R. and Schmidt, A. 2005. Adding usability to Web engineering models and tools. In *5th International Conference on Web Engineering (ICWE 2005)*, Sydney, Australia, Springer, New York, pp. 36–41.

Bastien, J.M. and Scapin, D.L. 1993. Ergonomic criteria for the evaluation of human-computer interfaces. Technical Report n.156. INRIA, Rocquencourt, France.

Blackmon, M.H., Polson, P.G., Kitajima, M., and Lewis, C. 2002. Cognitive walkthrough for the Web. In *Proceedings of the ACM CHI'02*, Minneapolis, MN, pp. 463–470.

Botafogo, R.A., Rivlin, E., and Shneiderman, B. 1992. Structural analysis of hypertexts: Identifying hierarchies and useful metrics. *ACM Transactions of Information Systems* 10(2): 142–180.

Calero, C., Ruiz, J., and Piattini, M. 2005. Classifying Web metrics using the Web quality model. *Online Information Review* 29(3): 227–248.

Ceri, S., Fraternali, P., and Bongio, A. 2000. Web modeling language (WebML): A Modeling language for designing Web sites. In *9th WWW Conference (2000)*, Amsterdam, the Netherlands, pp. 137–157.

Cockton, G., Lavery, D., and Woolrychn, A. 2003. Inspection-based evaluations. In Jacko, J.A. and Sears, A. (Eds.), *The Human-Computer Interaction Handbook*, 2nd edn., L. Erlbaum Associates, Hillsdale, NJ, pp. 1171–1190.

Conte, T., Massollar, J., Mendes, E., and Travassos, G.H. 2007. Usability evaluation based on Web design perspectives. In *1st International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, pp. 146–155, Spain.

Fernandez, A., Insfran, E., and Abrahão, S. 2009. Integrating a usability model into a model-driven web development process. In *10th International Conference on Web Information Systems Engineering (WISE 2009)*, Poznan, Poland, pp. 497–510, Springer-Verlag, New York.

Fernandez, A., Insfran, E., and Abrahão, S. 2011. Usability evaluation methods for the web: A systematic mapping study. *Information & Software Technology* 53(8): 789–817.

Gómez, J., Cachero, C., and Pastor, O. 2001. Conceptual modeling of device-independent Web applications. *IEEE Multimedia* 8(2): 26–39.

Hall, A. and Chapman, R. 2002. Correctness by construction: Developing a commercial secure system. *IEEE Software* 19(1): 18–25.

Hwang, W. and Salvendy, G. 2010. Number of people required for usability evaluation: The 10 ± 2 rule. *Communications of the ACM* 53(5): 130–133.

ISO/IEC 14598. 1999. Information technology, Software product evaluation.

ISO/IEC 9126-1. 2001. Software engineering, Product quality—Part 1: Quality model.

ISO/IEC 25010. 2011. Systems and software engineering—Systems and software quality requirements and evaluation (SQuaRE)—System and software quality models.

Ivory, M. Y. and Hearst, M. A. 2002. Improving Web site design. *IEEE Internet Computing* 6(2): 56–63.

Juristo, N., Moreno, A., and Sanchez-Segura, M.I. 2007. Guidelines for eliciting usability functionalities. *IEEE Transactions on Software Engineering* 33(11): 744–758.

Leavit, M. and Shneiderman, B. 2006. Research-based Web design & usability guidelines. U.S. Government Printing Office, Washington, DC. http://usability.gov/guidelines/index.html

Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., and López-Vaquero, V. 2004. USIXML: A language supporting multi-path development of user interfaces. In Bastide, R., Palanque, P. A., and Roth, J. (Eds.), *EHCI/DS-VIS*, Vol. 3425 of LNCS, Springer, New York, pp. 200–220.

Lynch, P. J. and Horton, S. 2002. *Web Style Guide: Basic Design Principles for Creating Web Sites*, 2nd edn., Yale University Press, London, U.K.

Molina, F. and Toval, J.A. 2009. 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.

Mori, G., Paterno, F., and Santoro, C. 2004. Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE TSE* 30(8): 507–520.

Nielsen, J. and Molich, R. 1990. Heuristic evaluation of user interfaces. In *Proceedings of ACM CHI'90 Conference*, Seattle, WA, pp. 249–256.

Norman, D.A. 1988. *The Psychology of Everyday Things*, Basic Books, New York.

Object Management Group (OMG). The MDA guide version 1.0.1. omg/2003–06–01. http://www.omg.org/cgi-bin/doc?omg/03–06–01 (accessed June 18, 2012).

Offutt, J. 2002. Quality attributes of Web software applications. *IEEE Software* 19(2): 25–32.

Olsina, L. and Rossi, G. 2002. Measuring Web application quality with WebQEM. *IEEE Multimedia* 9(4): 20–29.

Panach, I., Condori, N., Valverde, F., Aquino, N., and Pastor, O. 2008. Understandability measurement in an early usability evaluation for model-driven development: An empirical study. In *International Empirical Software Engineering and Measurement (ESEM 2008)*, Kaiserslautern, Germany, pp. 354–356.

Steuer, J. 1992. Defining virtual reality: Dimensions determining telepresence. *Journal of Communication* 42(4): 73–93.

Web Content Accessibility Guidelines 2.0 (WCAG 2.0). 2008. In Caldwell, B., Cooper, M., Guarino Reid, L., Vanderheiden, G. (Eds.), www.w3.org/TR/WCAG20 (accessed June 18, 2012).