Designing Highly Usable Web Applications

Silvia Abrahão, Emilio Insfran and Adrian Fernandez

Department of Information Systems and Computation
Universitat Politècnica de València
Camí de Vera s/n, 46022, Valencia, Spain
{sabrahao, einsfran, afernandez}@dsic.upv.es

43.1 Introduction

43.2 Underlying Principles

43.3 State of the Art in Usability Evaluation for Web Applications

43.4 Early Usability Evaluation in Model-Driven Web Development

43.5 Web Usability Model
   43.5.1 Usability Attributes
   43.5.2 Usability Metrics

43.6 Operationalizing the Web Usability Model
   43.6.1 Selecting the Web Development Process
   43.6.2 Applying the Web Usability Model in Practice

43.7 Summary

43.8 Future Research Opportunities

Defining Terms

References
43.1 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 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.

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 etc.) 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.

This MDE approach 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 this type of processes. This chapter addresses this issue by showing how usability evaluations are integrated into a specific model-driven development method to assure the usability of the Web applications developed.

43.2 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 input may not ensure the usability of the final Web application.

This problem may be alleviated by using a model-driven engineering approach due to its intrinsic traceability mechanisms that are established by the transformation processes. The most well-known approach to model driven engineering is the Model Driven Architecture (MDA) defined by the Object Management Group – OMG (2003). In MDA, platform-independent models (PIM) such as task models or navigation models may be transformed into platform-specific models (PSM) that contain specific implementation details from the underlying technology platform. These platform-specific models may then be used to generate the source code for the Web application (Code Model – CM), thus preserving the traceability among platform-independent models, platform-specific models, 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 (platform-independent or platform-specific models in this context) can be complemented with other evaluation methods performed with end-users before releasing a Web application to the public.

43.3 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 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 (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). 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 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.

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 (Ivyry and Hearst 2002) and 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 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 (2001) standard, 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 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 user interfaces. This approach takes advantage of models that represent the navigation (how the Website is traversed), and the user
interface (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 sub-characteristics 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 which 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.

43.4 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., 20014)) and languages (e.g., UsiXML (Limbourg et al. 2004)) that exploit model-driven engineering 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** which contains a set of usability attributes and metrics that can be applied by the Web designer in the following phases of a MDA-based Web development process: i) in the 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); ii) in the PSM, to inspect the concrete design models related to a specific platform; and iii) in the code model, to inspect the user interface generated or the Web application source code (see Figure 36.1).

![Diagram](image)

**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 only can be evaluated 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 the PIM (Figure 36.1; 1A). Changes in the PIM are reflected in the code model (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 the PSM produce a platform-specific usability report. If the 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 the 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.

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.

43.5 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.

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

43.5.1 Usability Attributes

SQuaRE decomposes usability into eight high-level sub-characteristics: appropriateness, recognisability, learnability, operability, user error protection, accessibility, user interface aesthetics, and compliance. However, these sub-characteristics are generic and need to be further broken down into measurable usability attributes. For this reason, our proposed Web Usability Model breaks down these sub-characteristics into other sub-characteristics 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 in Bastien and Scapin (1993) and other usability guidelines for Web development (Lynch and Horton 2002) (Leavit and Shneiderman 2006).

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

**Appropriateness recognisability** refers to the degree to which users can recognize whether a Web application is appropriate for their needs. In our Web Usability Model, this sub-characteristic 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 sub-characteristics such as familiarity, the ease with which a user recognizes the user interface 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 sub-characteristic was broken down in other sub-characteristics 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 which has been employed in the Human-Computer Interaction field in order to determine how intuitive the interaction is (Norman 1988). These sub-characteristics 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 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 sub-characteristic was broken down into other sub-characteristics 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 sub-characteristic 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 sub-characteristic was broken down in other sub-characteristics related to the 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 sub-characteristic as an attempt to integrate usability and accessibility issues. In our model, this sub-characteristic 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 etc.). 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 sub-characteristics of the usability criteria are related to the perception of the end-user (user interface 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 a user interface 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 sub-characteristic is clearly subjective and can be influenced by many factors in a specific use context, it is possible to define attributes which may have a high impact on how users perceive the Web application.

In our model, this sub-characteristic was broken down into other sub-characteristics related to the uniformity of the elements presented in the user interface (e.g., font, color, position), interface appearance customizability, which should not be confused with the sub-characteristic ‘capability of adaption’, since it is 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 sub-characteristic was broken down in other sub-characteristics such as the degree of fulfillment of the ISO/IEC 25010 (2011) standard and other relevant usability and Web design guidelines.

43.5.2 Usability Metrics

Once the sub-characteristics 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
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 etc.). If the Web metric was applied to the final user interface (CM), we also analyzed the possibility of adapting it to be applied at 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.
16

TABLE 36.1 Examples of Usability Measures

<table>
<thead>
<tr>
<th>Subcharacteristic/Usability Attribute</th>
<th>Measure</th>
<th>Definition</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learnability/Predictability/Icon/Link Title Significance</td>
<td>Proportion of titles chosen suitably for each icon/title</td>
<td>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: $0 &lt; X \leq 1$)</td>
<td>PIM, PSM, or CM</td>
</tr>
<tr>
<td>Appropriateness recognizability/ User guidance/Quality of messages</td>
<td>Proportion of meaningful messages (error, advise, and warning messages)</td>
<td>Ratio of error messages explaining an error in a proper way and the total contexts where this error may occur (scale: $0 &lt; X \leq 1$)</td>
<td>PIM, PSM, or CM</td>
</tr>
<tr>
<td>Appropriateness recognizability/ Navigability/Reachability</td>
<td>Breadth of the internavigation (BiN)</td>
<td>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 &gt; 0)</td>
<td>PIM</td>
</tr>
<tr>
<td></td>
<td>Depth of the navigation (DN)</td>
<td>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 &gt; 0)</td>
<td>PIM</td>
</tr>
</tbody>
</table>

43.6 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 platform-independent and platform-
specific models (PIMs and PSMs) of the Web development method.

43.6.1 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 platform-independent models (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 user interface. 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.

43.6.2 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) functions. Figure 36.2 (a) shows a fragment of the Navigational Access Diagram that represents the user access to the application whereas Figure 36.2 (b) shows the Abstract Presentation Model generated from the Class Model and the Navigational Model.

The Navigational Access Diagram (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 the user interface. 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 36.2 (a), the Navigational Access Diagram 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 36.2 (b), the Abstract Presentation Model shows the three abstract pages generated from the previous Navigational Access Diagram. The User navigational class produces a Web page showing the abstract access to the application (see Figure 36.2; b-1). The navigational link LI2 generates the error message represented by the error collection in a different abstract Web page (see Figure 36.2; b-2). 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; b-3) 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 login information is showed in the same user’s generated home page (see Figure 36.2; b-3).

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

- **Breadth of the inter-navigation** (*see Table 36.1*): This metric can only be operationalized in those NADs that represent the first navigation level (i.e., inter-navigation). The calculation formula to be applied is: 
  
  \[
  BiN(NAD) = \text{Number of Output Target Links from Collections connected to Navigational targets}
  \]

  The thresholds defined for the metric are:
  
  - \([BiN = 0]\): critical usability problem;
  - \([1 \leq BiN \leq 9]\): no usability problem;
  - \([10 \leq BiN \leq 14]\): low usability problem;
  - \([15 \leq BiN \leq 19]\): medium usability problem.
  
  Therefore, the breadth of the inter-navigation for the NAD shown in Figure 36.2 (a) is 4 since there are 4 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:

  \[
  DN(NAD) = \text{Number of navigation steps from the longest navigation path},
  \]

  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\(^1\) defined for this metric are: \([1 \leq DN \leq 4]\): no usability problem; \([5 \leq DN \leq 7]\): low usability problem; \([8 \leq DN \leq 10]\): medium usability problem; \([DN \geq 10]\): critical usability problem. Therefore, the depth of the navigation for the NAD shown in Figure 36.2 (a) is 2 because the length of the longest navigational path between the root of the navigational map and its leaves (Tasks, Contacts, and navigational targets Reports) passes through two navigational links. This means that no usability problem was detected since obtained values are within the threshold \([1 \leq BaN \leq 9]\).

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 \textit{depth} of less than 5 levels and an \textit{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 36.2 (b), the following metrics, which are closer to the final user interface, may be applied:

- \textit{Proportion of meaningful messages} (see Table 36.1): There is only one error message “Login 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

-----

\(^1\) These thresholds were established considering Hypertext research works such as Botafogo et al. (1992), and usability guidelines such as Leavit and Shneiderman (2006) and Lynch and Horton (2002).
closer to 1 indicate that the error messages are very significant, thus guiding the user towards 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 36.2 (b): *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 model driven engineering approach. Although the usability model has been operationalized in OO-H, it can also be used with other model-driven Web development methods e.g., 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 platform-independent models, platform-specific models 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.

43.7 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 Software Product Quality Requirements and Evaluation (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 at 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 improvement recommendations).

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 during the maintenance phase. This is in line with research efforts in the Computer Science community towards 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, but 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 in consideration when performing usability inspections. This may lead us to determine 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.

4.3.8 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 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 review that we performed to investigate which usability evaluation methods have been used to evaluate Web artifacts and how they were employed [13]. 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.
Defining Terms

**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 viewpoint. 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 final code generation.

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

References


Conf. Seattle, WA, 249-256.


