5 Web Usability: Principles and Evaluation Methods

Maristella Matera, Francesca Rizzo, Giovanni Toffetti Carughi

Abstract: Current Web applications are very complex and highly sophisticated software products, whose usability can greatly determine their success or failure. Defining methods for ensuring usability is one of the current goals of Web engineering research. Also, much attention is currently paid to usability by industry, recognising the importance of adopting methods for usability evaluation before and after application deployment. This chapter introduces principles and evaluation methods to be adopted during the whole application lifecycle for promoting usability. For each evaluation method, the main features, as well as the emerging advantages and drawbacks, are illustrated so as to support the choice of an evaluation plan that best fits the goals to be pursued and the available resources. The design and evaluation of a real application is also described for exemplifying the concepts and methods introduced.

Keywords: Web usability, Evaluation methods, Web usability principles, Development process.

5.1 Introduction

The World Wide Web has had a significant impact on access to the large quantity of information available through the Internet. Web-based applications have influenced several domains, by providing access to information and services to a variety of users with different characteristics and backgrounds. Users visit Web applications, and return to previously accessed applications, if they can easily find useful information, organised in a way that facilitates access and navigation, and presented according to a well-structured layout. In other words, the acceptability of Web applications by users relies strictly on the applications' *usability*.

Usability is one relevant factor of a Web application's quality. Recently, it has received great attention, and been recognised as a fundamental property for the success of Web applications. Defining methods for ensuring usability is therefore one of the current goals of Web engineering research. Also, much attention is currently paid to usability by industry, which is recognising the importance of adopting usability methods during the development process, to verify the usability of Web applications before and

after their deployment. Some studies have demonstrated how the use of such methods reduces costs, with a high cost benefit ratio, as they reduce the need for changes after the application is delivered [40,50].

5.1.1 Usability in the Software Lifecycle

Traditional software engineering processes do not explicitly address usability within their lifecycles. They suggest different activities, from the initial inception of an idea until the product deployment, where testing is conducted at the end of the cycle to check if the application design satisfies the high-level requirements, agreed by the customer, is complete and internally consistent. To achieve usable applications, it is necessary to extend the standard lifecycle to explicitly address usability issues. This objective does not imply simply adding some activities; rather it requires appropriate techniques which span the entire lifecycle [20].

Given the emergent need for usability, traditional development processes were extended to enable the fulfilment of usability requirements. Evaluation methods have been adopted at all stages within the process, to verify the usability of incremental design artefacts, as well as of the final product. This has resulted in the proposal of the so-called *iterative design* [58,16] for promoting usability throughout the entire development lifecycle.

With respect to more traditional approaches, which suggest the use of a top-down method (such as for example the waterfall model), iterative design prescribes that the development process be complemented by a bottom-up, synthetic approach, in which the requirements, the design, and the product gradually evolve to become well defined. The essence of iterative design is that the only way to be sure about the effectiveness of design decisions is by building and evaluating application prototypes. The design can then be modified, to correct any false assumptions detected during the evaluation activities, or to accommodate new requirements; the cycle represented by design, evaluation, and redesign must be repeated as often as necessary.

In this context, *usability evaluation* is interpreted as an extension of testing, carried out through the use of prototypes with the aim of verifying the application design against usability requirements. Evaluation is central to this model: it is relevant at all the stages in the lifecycle, not just at the end of the product development. All aspects of the application development are in fact subject to constant evaluation, involving expert evaluators and users.

Iterative development is consistent with the real nature of design. It emphasises the role of prototyping and evaluation, the discovery of new requirements, and the importance of involving diverse stakeholders – including users.

What makes iterative development more than merely well-intentioned trial and error? Usability engineering became the banner under which diverse methodological endeavours were carried throughout the 1990s:

- It proposes that iterative development is managed according to explicit and measurable objectives, called "usability specifications", which must be identified early in the development process. Explicit usability goals are therefore incorporated within the design process, emphasising that the least expensive way of obtaining usable products is to consider usability issues early in the lifecycle, reducing the need to modify the design at the end of the process [44,45].
- It suggests the use of "simple usability engineering", which adopts easy- to-apply, and efficient, evaluation techniques, encouraging developers to consider usability issues throughout the whole development cycle [47].

5.1.2 Chapter Organisation

The aim of this chapter is to illustrate usability principles and evaluation methods that, in the context of an iterative design process, can support the production of usable Web applications. After introducing the general concept of usability and its specialisation for the Web, we present usability criteria that support Web usability in two ways: first, they can guide the design process, providing guidelines on how to organise the application by means of usable solutions; second, they drive the evaluation process, providing benchmarks for usability assessment. We will then present evaluation methods to be tackled during the entire development process – both during design and after application deployment based on the intervention of usability specialists, or involvement of real users.

In order to exemplify the concepts introduced, we discuss several important usability issues during the design and evaluation of a real Web application, developed for the Department of Electronics and Information (DEI) at Politecnico di Milano (http://www.elet.polimi.it). The DEI application is a very large, data-intensive application, consisting of:

- A *public area*, publishing information about the Department staff, and their teaching and research activities. It receives about 9000 page requests per day from external users.
- An *intranet area*, supporting some administrative tasks available to 300 DEI members.
- A *content management area*, which provides Web administrators with an easy-to-use user interface front-end for creating or updating content to be published via the Web application.

5.2 Defining Web Usability

Usability is generally taken as a software quality factor that aims to provide the answer to many frustrating problems caused by the interaction between people and technology. It describes the quality of products and systems from the point of view of its users.

Different definitions of usability have been proposed, which vary according to the models they are based on. Part 11 of the international standard ISO 9241 (*Ergonomic Requirements for Office Work with Visual Display Terminals*) provides guidance on usability, introducing requirements and recommendations to be used during application design and evaluation [29]. The standard defines usability as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". In this definition, *effectiveness* means "the accuracy and completeness with which users achieve specified goals", *efficiency* is "the resources expended in relation to the accuracy and completeness with which users achieve goals", and *satisfaction* is described as "the comfort and acceptability of use". Usability problems therefore refer to aspects that make the application ineffective, inefficient, and difficult to learn and use.

Although the ISO 9241-11 recommendations have become the standard for the usability specialists' community, the usability definition most widely adopted is the one introduced by Nielsen [45]. It provides a detailed model in terms of usability constituents that are suitable to be objectively and empirically verified through different evaluation methods. According to Nielsen's definition, usability refers to:

- *Learnability*: the ease of learning the functionality and behaviour of the system.
- *Efficiency*: the level of attainable productivity, once the user has learned the system.
- *Memorability*: the ease of remembering the system functionality, so that the casual user can return to the system after a period of non-use, without needing to learn again how to use it.
- *Few errors*: the capability of the system to feature a low error rate, to support users making few errors during the use of the system, and, in case they make errors, to help them recover easily.
- *Users' satisfaction*: the measure in which the user finds the system pleasant to use.

The previous principles can be further specialised and decomposed into finer-grained criteria that can be verified through different evaluation methods. The resulting advantage is that more precise and measurable criteria

contribute towards setting an engineering discipline, where usability is not just argued, but systematically approached, evaluated, and improved [44,45].

When applying usability to Web applications, refinements need to be applied to the general definitions, to capture the specificity of this application class. Main tasks for the Web include: finding desired information and services by direct search, or the discovery of others by browsing; comprehending the information presented; invoking and executing services specific to certain Web applications, such as the ordering and downloading of products. Paraphrasing the ISO definition, Web usability can therefore be considered as the ability of Web applications to support such tasks with effectiveness, efficiency, and satisfaction. Also, Nielsen's usability principles mentioned above can be interpreted as follows [48]:

- *Web application learnability* must be interpreted as the ease for Web users to understand the contents and services made available through the application, and how to look for specific information using the available links for hypertext browsing. Learnability also means that each page in the hypertext front-end should be composed in a way such that its contents are easy to understand and navigational mechanisms are easy to identify.
- *Web applications efficiency* means that any content can be easily reached by users through available links. Also, when users get to a page, they must be able to orient themselves and understand the meaning of this page with respect to the starting point of their navigation.
- *Memorability* implies that, after a period of non-use, users are still able to orient themselves within the hypertext; for example, by means of navigation bars pointing to landmark pages.
- *Few errors* mean that when users erroneously follow a link, they are able to return to their previous location.
- *Users' satisfaction* refers to the situation in which users feel they are in control with respect to the hypertext, since they comprehend the available content and navigational commands.

In order to be evaluated, the previous criteria can be further refined into more objective and measurable criteria. Section 5.3 will introduce a set of operational criteria for Web application design and evaluation.

5.2.1 Usability and Accessibility

Recently, the concept of usability has been extended to include *accessibility*. Accessibility focuses on application features that support universal access by any class of users and technology [59]. In particular, accessibility

focuses on properties of the mark-up code that make page contents "readable" by technologies assisting impaired users. Some literature gives *accessibility* a broader meaning: that is, the ability of an application to support any users identifying, retrieving, and navigating its contents [26,63]. In fact, accessible Web applications are advantageous to any users, especially in specific contexts of use, such as adopting voice-based devices (e.g. cellular phones) while driving. According to this meaning, accessibility can therefore be considered a particular facet of Web usability.

The W3C Web Accessibility Initiative (WAI) acts as the central point for setting accessibility guidelines for the Web. Its work concentrates on the production of Web Content Accessibility Guidelines (WCAG 2.0) [72], which focus on two main goals:

- *Producing contents that must be perceivable and operable*: this implies using a simple and clear language, as well as defining navigation and orientation mechanisms for supporting content access and browsing.
- *Ensuring access alternatives*: this means that pages must be designed and coded so they can be accessed independently from the adopted browsing technologies and devices, and from the usage environment.

The first goal is strictly related to the definition of Web usability; it can be pursued by focusing on usability criteria that enhance the effectiveness and efficiency of navigation and orientation mechanisms. The second goal can be achieved via the page mark-up, and in particular:

- *Separating presentation from content and navigation design*, which enables an application to present the same content and navigational commands according to multiple presentation modalities, suitable for different devices.
- *Augmenting multimedia content with textual descriptions*, so it can be presented through alternative browsing technologies, such as screen readers for assisting impaired users.
- *Creating documents that can be accessed by different types of hardware devices*. For example, it should be possible to interact with page contents even through voice devices, small-size devices, or black and white screens, and when pointing devices are not available.

WCAG recommendations provide 14 guidelines, each specifying how it can be applied within a specific context. For further details the reader is referred to [72].

5.3 Web Usability Criteria

According to the usability engineering approach, a cost-effective way to increase usability is for it to be addressed from the early phases of an application's development. A solution for achieving this goal is to take into account criteria that refine general usability principles (such as those presented in Sect. 5.2), suggesting how the application must be organised to conform to usability requirements [45]. Such criteria drive the design activity, providing guidelines on how to restrict the space of design alternatives, thus preventing designers from adopting solutions that can lead to unusable applications [20]. In addition, they constitute the background for the evaluation activity.

The development of Web applications, according to several methods recently introduced in Web engineering [5,14,57], must focus on three separate dimensions: data, hypertext, and presentation design - each being accompanied by a set criterion. Criteria so far proposed for the design of user interfaces [28,45,53], as well as the W3C-WCAG guidelines for accessibility, work well for organising the presentation layer of Web applications [39,49]. Table 5.1 summarises the ten "golden rules" proposed by Nielsen in 1993 for the design and evaluation of interactive systems.

More specific criteria are, however, needed for addressing the specific requirements, conventions and constraints characteristic of the design of content and hypertext links in Web applications. This section therefore proposes a set of criteria that suggest how Web applications should be organised, at the data and hypertext level, supporting information finding, browsing, and user orientation. These represent the three fundamental aspects we believe have the greatest impact on usability of Web applications. The criteria have been defined in the context of a model-driven design method [14,15]; as such, they take advantage of adopting a few high-level conceptual abstractions for systematically planning the overall structure of the application, avoiding implementation details and mark-up coding of single pages. Our method focuses on the broad organisation of the information content and the hypertext structure ("in-the-large"). In particular, the criteria are based on the assumption that the retrieval and fruition of content by end users is significantly affected by the way in which the content itself is conceived, designed, and later delivered by the hypertext interface. This assumption is also supported by a recommendation coming from the fields of human computer interaction and human factor studies [41,69,70].

Table 5.1. Nielsen's ten heuristics for user interface design and evaluation (http://www.useit.com/papers/heuristic/heuristic_list.html)

	HEURISTIC	DESCRIPTION
1.	Visibility of system status	The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
2.	Match between system and the real world	The system should speak the users' language, with words, phrases, and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
3.	User control and freedom	Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
4.	Consistency and standards	Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
5.	Error prevention	Even better than good error messages is a careful design which prevents a problem from occurring in the first place.
6.	Recognition rather than recall	Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
7.	Flexibility and efficiency of use	Accelerators - unseen by the novice user - may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
8.	Aesthetic and minimalist design	Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
9.	Help users recognise, diagnose, and recover from errors	Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
10.	Help and documentation	Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large

The usability of Web applications thus requires the complete understanding and accurate modelling of data resources. As such, and differently from previous proposals [25,48,49], our criteria are organised as general principles later expanded into two sets of more practical guidelines, one suggesting how to structure content, and another proposing the definition of usable navigation and orientation mechanisms for content access and browsing.

5.3.1 Content Visibility

In order to understand the structure of the information offered by the application, and become oriented within hypertext, users must be able to easily identify the main conceptual classes of contents.

Identification of Core Information Concepts

Content visibility can be supported by an appropriate content design, where the main classes of content are identified and adequately structured. To fulfil this requirement, the application design should start from the identification of the information entities modelling the *core concepts* of the application, which act as the application backbones, representing the best answer to users' information requirements [15]. Data design will be centred on such content, and will gradually evolve by detailing its structure in terms of elementary components, and further add access and browsing content.

Hypertext Modularity

The hypertext must be designed to support users to *perceive* where core concepts are located. To this end:

- The hypertext can be organised in *areas,* i.e. modularisation constructs grouping pages that publish homogeneous contents. Each one should refer to a given core concept identified at a data level.
- Areas must be defined as *global landmarks* accessible through links, grouped in *global navigation bars* that are displayed in any page of the application interface.
- Within each area, the most representative pages (e.g. the area entry page, search pages, or any other page from which users can invoke relevant operations) can be defined as *local landmarks*, reachable through *local navigation bars* displayed in any page within the area.

These links supply users with cornerstones to enhance their orientation within the area.

The regular use of hierarchical landmarks within pages enhances learnability and memorability: landmarks indeed provide intuitive mechanisms for highlighting the available content and the location within the hypertext where they are placed. Once learned, they also support orientation and error recovery, as they are available throughout the application as the simplest mechanism for context change.

Content Visibility in the DEI Application

In the DEI application, the core concepts of the public module are the *research areas*, the *teaching activities*, the *industrial projects*, and the *DEI members.* In accordance with this organisation of information content, the hypertext of the Web application is organised into four areas, *Research, Teaching, Industry, and People*, each corresponding to a single core concept (see Fig. 5.1).

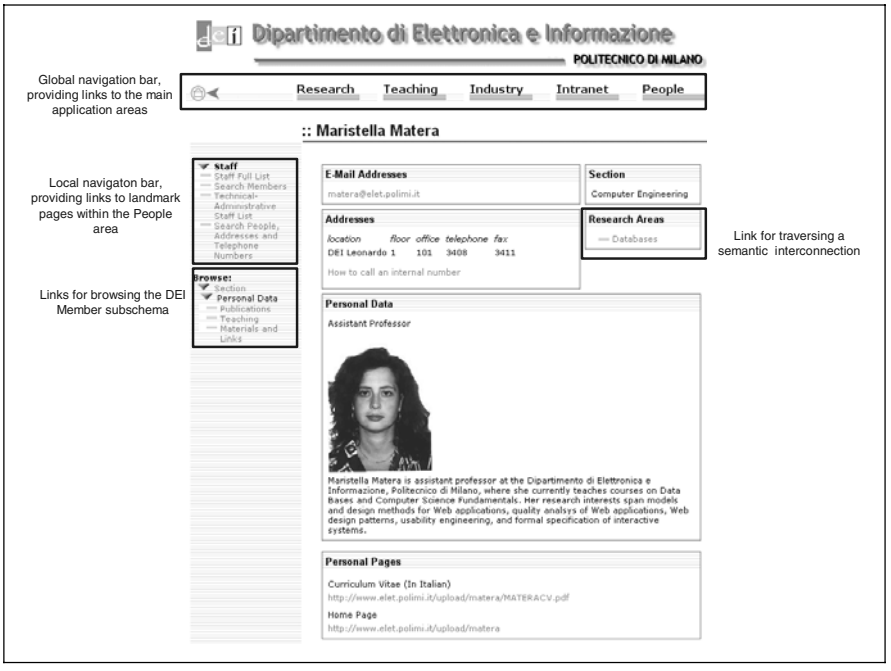

Fig. 5.1. Page organisation, with global and local landmarks, and core, peripheral, and interconnection sections

Each page within the application includes a global navigation bar, grouping links to the four application areas. Also, each page contains a local navigation bar that groups links to local landmarks. Figure 5.1 shows a page from the *People* area, which displays information about a DEI member. The global navigation bar is placed in the top region of the page. The bar also includes a link to the non-public intranet and to the Home Page. Landmarks defined locally for the *People* area are placed in the top region of the left-side bar.

5.3.2 Ease of Content Access

Once users have identified the application's main content classes, they must be provided with "facilities" for accessing the specific content items they are interested in.

Identification of Access Information Concepts

The design of access paths for retrieving core content items can be facilitated if designers augment the application content with *access concepts*, which correspond to classification criteria or context over core concepts. These enable users to move progressively from broader to narrower categories, until they locate the specific core concept of interest [49]. In general, multiple and orthogonal hierarchies of access concepts should be related to every core concept.

Navigational Access and Search-Based Access

In order to facilitate access to specific instances of core concepts, access concepts, defined at data level, should be used to construct *navigational access mechanisms*. These typically consist of multi-level indexes, possibly distributed on several *access pages*, bridging pages with a high visibility (e.g. the Home Page or the entry page of each area), to pages devoted to the publication of core concepts.

Especially in large Web applications, navigational access is often complemented with *direct access*, i.e. keyword-based search mechanisms, which allow users to avoid navigation and to rapidly reach the desired information objects. Direct access mechanisms are essential for interfaces (such as those of mobile devices) that are not able to support multiple navigation steps. In traditional hypertext interfaces, they enhance orientation when users "get lost" while moving along navigational access mechanisms [60,49].

Pairing navigational and direct access with explicit visibility over available categorisations and free text queries, in addition to a regular use of these access mechanisms within the hypertext, can greatly enhance content accessibility.

Content Access in the DEI Application

In the DEI application, each area is provided with navigational and direct access. Figure 5.2 shows the contextual access path defined for the core concept *DEI Member*. It consists of a hierarchy of indexes, developed through different access pages, which let users move from broader *People* categories, presented in the application's Home Page (e.g. academic staff), to pages listing narrower sub-categories (e.g. all the categories of the academic staff). In addition, users can move to the list of members in a selected sub-category, from which they can select a person's name and access her/his corresponding page. Each page also provides direct access, by means of a keyword-based search, for directly reaching single DEI members.

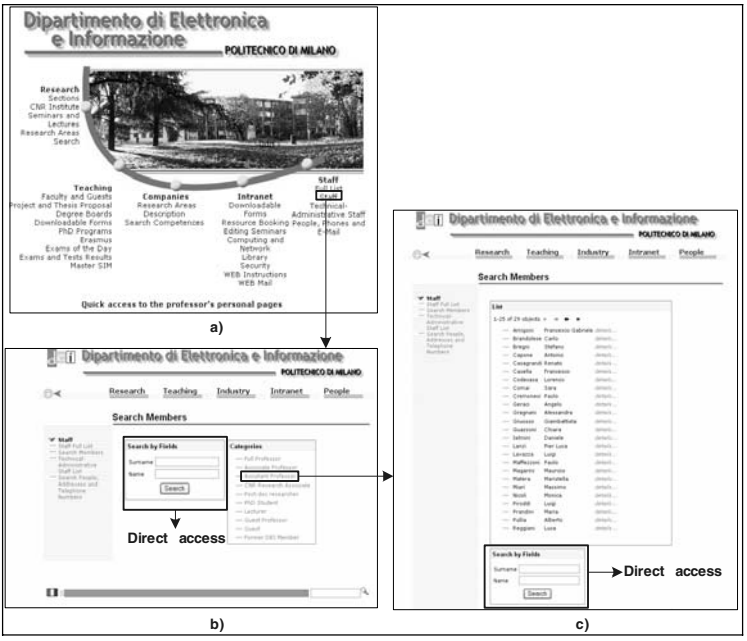

Fig. 5.2. Hierarchy of indexes in the navigational access to the DEI Member, consisting of the Home Page (a), the Member Categories page (b), and the Category Member Index page (c). Pages (b) and (c) also include a keyword-based search for direct access

5.3.3 Ease of Content Browsing

Users must be able to easily identify possible auxiliary content related to each single core concept, as well as the available interconnections between different core concepts.

Core Concepts' Structuring and Interconnection

The ease of use and learnability of a Web application can be enhanced by supporting users' understanding of the content structure and the semantic interconnections defined between different content classes. Therefore, when the core concepts represent a structured and complex concept, it is recommended that they be expanded, using a top-down design, into a composite data structure. Such structure collectively represents the entire core concept, and is characterised by:

- A *central information content* – which expresses the concept's main content and provides to an individual the means to identity each core concept.
- Some other *peripheral information elements*, which complete the concept's description.

Semantic interconnections among core concepts must be established for producing a knowledge network through which users can easily move, and explore the information content [41]. If defined, interconnections allow users to comprehend a Web application's structure and how to navigate through it efficiently.

Organisation of Core Pages

In order to highlight the structure of each core concept, and the interconnections between different concepts, pages devoted to core concept presentation should contain at least three sections:

- A *core section* that clearly conveys the content associated with the core concept.
- A *peripheral* section that highlights auxiliary information – if any – completing the core concept.
- An *interconnection* section that represents links to other pages within the area, or to the core contents of other areas.

The definition of the internal structure of pages by means of these three sections facilitates the understanding of the information in the page. If systematically repeated through the application, it enhances consistency among the components displayed by pages [49]. In addition, it is perceived

as a key component in helping users understand the application's hypertextual structure, and to support a conscious shift of focus by users. Finally, if the structure is explicitly annotated on the page mark-up code, it can be used to build intelligent page readers, and thus enable accessibility to any users.

Content Browsing in the DEI Application

As an example of the organisation of content browsing in the DEI application, let us consider the DEI Member page (see Fig. 5.1). The page features as a core section a central region that presents the attributes qualifying a DEI Member (e-mail address, postal address, department section, biography, list of personal pages). The page then includes two link groups:

- The first refers to the page's peripheral section, and points to pages that contain further details about each DEI member (e.g. the list of publications and courses).
- The second represents the page's interconnection section, thus enabling a semantic interconnection, and points to the research area the member belongs to.

Note that such page structure also applies to pages in other application areas.

5.4 Evaluation Methods

Applying principles for the design of usable applications is not sufficient to ensure good usability of the final product. Even though systematic design techniques can be used, it is still necessary to check the intermediate results, and to test the final application to verify if it actually shows the expected features, and meets the user requirements. The role of *evaluation* is to help verify such issues.

The three main goals of an evaluation are, first, to assess the application's functionality; second, to verify the effect of the application's interface on the user; third, to identify any specific problems with the application, such as aspects which show unexpected effects when used within the intended context [20]. In relation to Web applications, an evaluation should verify if the application design allows users to easily retrieve and browse content, and to invoke available services and operations. Therefore, it implies not only to have the appropriate content and services available, but also to make them easily reachable to users by means of adequate hypertext structures.

Depending on the phase in which an evaluation is performed, it is possible to distinguish between *formative evaluation*, which takes place during the design stage, and *summative evaluation*, which takes place after the product has been developed, or when a prototype is ready. During the early design stages, the goal of a formative evaluation is to provide feedback during the design activities by checking the design team's understanding of the users' requirements, and by testing design choices quickly and informally. Later, a summative evaluation can be used to identify users' difficulties using the application, and help improve the final product or prototype.

Within these two broad categories, there are different methods that can be used at different stages of the development cycle of an application. The most commonly adopted methods are *user testing*, where the real users participate, and *usability inspection*, which is conducted by specialists. Recently, *Web usage analysis* has also emerged as a method for studying user behaviour through the computation of access statistics, and the reconstruction of user navigation on the basis of Web access logs.

The remainder of this section illustrates the main features of these three classes of evaluation methods, and also highlights their advantages and drawbacks.

5.4.1 User Testing

User testing aims to investigate real users' behaviour, observed using a representative sample of real users [46]. It requires users to perform a set of tasks using physical artefacts, which can be either prototypes or finished applications, while an investigator observes their behaviour and gathers data about the way users execute assigned tasks [20,55,68]. In general the data gathered during such investigations are user's execution time, number of errors, and user satisfaction. After the user test is complete, the collected data are analysed and used to improve the application's usability.

Usability testing is explicitly devoted to analysing in detail how users interact with the application while accomplishing well-defined tasks. This characteristic differentiates between usability and beta testing, which is largely applied in industry. Beta testing is always carried out using the final product, where after an application's release, end users are asked about their satisfaction with the product. Conversely, usability testing is conducted by observing a sample of users that perform specific tasks while interacting with the application. The test is usually video recorded. The list of detected problems is reported, in addition to specific redesign suggestions.

To avoid unreliable and biased results, the design of a user test evaluation and its execution should be carefully planned and managed. A good usability test should involve the following steps:

1. *Define the goals of the test*. The objective of the evaluation can be generic (e.g. to improve end users' satisfaction with and the design of a product); or it can be specific (e.g. to evaluate the effectiveness of a navigational bar for user orientation, or the readability of labels).
2. *Define the user sample to participate in the test*. The user sample for the test should be representative of the population of end users that will use the application or prototype under scrutiny. Failing to do so will provide results that cannot be generalised to the population of real users. Possible criteria to use to define the sample are: user's experience (experts vs. novices), age, application's frequency of use, and experience with similar applications. The number of participants can vary, depending on the objectives of the test. Nielsen and Molich [52] assert that 50% of the most important usability problems can be identified with three users. Other authors claim that five users enable the discovery of 90% of usability problems [47,64]. Note that the use of very small samples not suggested by the literature on empirical investigations: thus, within the context of this book, they are simply informative.
3. *Select tasks and scenarios*. The tasks to be carried out during the test have to be real, i.e. they have to represent the activities people would normally perform with the application. Task scenarios can be obtained from the requirements phase. In addition, tasks can also be intentionally prepared to test unexpected situations.
4. *Define how to measure usability*. Before conducting a usability test, it is important to define the attributes that will be used to measure the results. Such attributes, or measures, can be qualitative[1] (e.g. user satisfaction, or difficulty of use), or quantitative (e.g. task completion time, number and typology of errors, number of successfully accomplished tasks, the amount of time users invoke help (verbal, on- line help, manual)). Users' anonymity should be guaranteed, and participants should also be provided with the test results. Besides observing, an investigator can also use other techniques for gathering data on task execution. Examples of such techniques are: the *think aloud* protocol, in which a subject is required to talk out loud while executing tasks, explaining the actions (s)he is trying to tackle, their reason, and the expectations; the *co-discovery* (or collaborative) approach, in which two participants execute the tasks together, helping each other; the *active intervention*, in which the investigator asks participants to reflect upon the events of the test session. It is worth noting that such techniques do not provide ways for measuring users' satisfaction. Such subjective measurement can instead be obtained through *survey techniques*, based on the use of questionnaires and interviews [35,58], to be answered by users after the completion of testing.

[1] Qualitative measures are also known to be subjective.

5. *Prepare the material and the experimental environment*. The experimental environment should be organised and equipped with a computer and a video camera for recording user activities. In addition, it is also important to establish the roles of the investigative team members, and prepare any supporting material (e.g. manuals, pencils, paper). Prior to running the test, a pilot trial is necessary to check, and possibly refine, all test procedures. Note that it is not mandatory to execute the test in a laboratory.

5.4.2 Inspection Methods

User testing is considered the most effective way of assessing the use of products and prototypes, from a real user's point of view. However, user testing is an expensive activity. In addition, to be useful, feedback needs to be obtained at earlier stages in the development process, and repeated throughout the process. Such constraints have led to the proposal of usability inspection methods, to be used by developers to predict usability problems that could be detected through user testing.

Usability inspection refers to a set of evaluation techniques that have evolved from inspection methods, used in software engineering, to debug and improve code. Within the context of usability, inspectors examine usability-related aspects of an application, to detect violations of established usability principles [51], and to provide feedback to designers about necessary design improvements. Such inspectors can be usability specialists, designers, and engineers with special expertise (e.g. knowledge of specific domains or standards). To be effective, inspection methods rely upon a good understanding of usability principles, how these principles affect the specific application being analysed, and the skills of the inspector to discover problems where the main violations occur.

Usability inspection methods were proposed as a cost-effective alternative to traditional usability evaluation [8]. The cost of user test studies and laboratory experiments became a central issue, and therefore many usability evaluation techniques were proposed, based on the involvement of specialists to supplement or even replace direct user testing [51,52].

Different methods can be used for inspecting an application [51]. The most commonly used method is *heuristic evaluation* [45,51], in which usability specialists judge if an application's properties conform to established usability principles. Another method is *cognitive walkthrough* [54,67], which uses detailed procedures for simulating users' problem-solving processes, to assess if the functions provided by the application are efficient for users, and can lead them to correct actions. The remainder of this section describes these two methods in more depth.

A detailed description of other inspection techniques is provided in [51].

Heuristic Evaluation

Heuristic evaluation is the most informal of inspection methods. It prescribes having a small set of experts analysing the application against a list of recognised usability principles – the heuristics. This technique is part of the so-called discount usability method. In fact, research has shown that it is a very efficient usability engineering method [32], with a high cost-benefit [47].

During the evaluation session, each evaluator goes through the system interface at least twice. The first step is to obtain an overall understanding of the flow of interaction and the general scope of the application. The second step focuses on specific objects and functionality, evaluating their design and implementation against a list of heuristics. The output of a heuristic evaluation session is a list of usability problems with reference to the violated heuristics (see Table 5.2 for an example). The reporting of problems caused by the violation of heuristics enables an easy generation of a revised design. The revised design is prepared in accordance with what is prescribed by the guidelines underlying the violated principles. Once the evaluation has been completed, the findings of the different evaluators are compared and aggregated.

Table 5.2. An example of table for reporting heuristic violations

Found problem	Violated heuristic	Severity	Suggested improvement
Download time is not indicated	Feedback	High	Use a scrolling bar for representing the time left till the end of download

Heuristic evaluations are especially valuable when time and resources are short, given that skilled evaluators can produce high-quality results in a limited amount of time, without the need for real users' involvement [34]. In principle, heuristic evaluation can be conducted by a single evaluator. However, in an analysis of six studies, it has been found that single evaluators are able to find only 35% of the total number of existing usability problems [43], and that different evaluators tend to find different problems. Therefore, it seems that the more experts involved in the evaluation, the greater the number of different problems that can be identified. Figure 5.3 shows the percentage of usability problems found by number of evaluators, as reflected by a mathematical model defined in [50]. The curve suggests that five evaluators may be able to identify close to 75% of usability problems; however, such results should be interpreted with caution, since they are reliant on the data from which they were obtained.

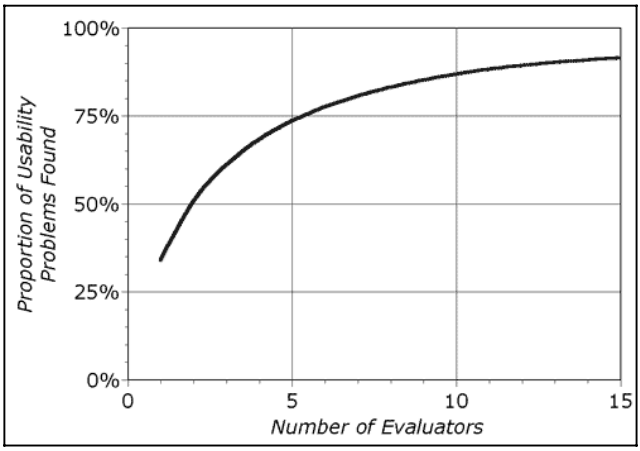

Fig. 5.3. The percentage of usability problems found by heuristic evaluation when using different numbers of evaluators [50]

Heuristic evaluations can have a number of drawbacks, with the major one being a high dependence on the skills and experience of the evaluators [21,33,34]. Nielsen states that novice evaluators with no usability expertise are poor evaluators, that usability experts are 1.8 times as good, and that application domain and usability experts are 2.7 times as good [44,45]. These results suggest that specific experience with a specific category of applications may significantly improve evaluators' performance.

Cognitive Walkthrough

A cognitive walkthrough simulates the user's problem-solving process, i.e. what the user will do in specific situations of use and why [54]. Evaluators go through the interface, step by step, using a task scenario, and discuss the usability issues as they arise. In particular, the method guides evaluators in the analysis of the actions that users would accomplish to reach the objectives defined in the scenario, by means of the identification of the relationship between user goals, actions, and the visible states of the application interface [27]. As such, cognitive walkthrough is particularly suited for the detection of problems affecting an application's learnability.

Cognitive walkthrough is a technique largely applied to evaluating aspects of an application's interface. Its use is recommended in the advanced phases of Web application development, to evaluate high-fidelity prototypes for which the interaction functionalities already work. The typical cognitive walkthrough procedure prescribes that, on the basis of selected scenarios of use, a series of tasks are chosen to be performed by an expert evaluator on the interface. The evaluator executes such tasks, and after the

completion of each elementary action (s)he interprets the application's answer, and evaluates the steps forward for the achievement of the end user's goal, by answering the following standard questions:

1. Are the feasible and correct actions sufficiently evident to the user, and do the actions match with her/his intention?
2. Will the user associate the correct action's description with what (s)he is trying to do?
3. Will the user receive feedback in the same place where (s)he has performed her/his action and in the same modality?
4. Does the user interpret the system's response correctly: does (s)he know if (s)he has made a right or wrong choice?
5. Does the user properly evaluate the results: is (s)he able to assess if (s)he got closer to her/his goal?
6. Does the user understand if the intention (s)he is trying to fulfil cannot be accomplished with the current state of the world: does (s)he find alternative goals?

During this interpretation process, it is also possible that the user/evaluator needs to change her/his initial goal because it is impossible to achieve. Each negative answer to the previous questions increments the list of detected problems. At the end of the evaluation session, the list of problems is completed with the indications of possible design amendments, and communicated back to the design team.

Web Usage Analysis

A recent direction in the evaluation of Web applications is called Web usage analysis [30]. It is performed using the recorded users' access to the application's Web pages, stored in a Web server log [61], according to one of the available standard formats [71]. This technique can only be used once a Web application is deployed, and can be used to analyse how users exploit and browse the information provided by the application. For instance, it can help discover navigation patterns that correspond to high Web usage, or those which correspond to early leaving.

Very often, Web logs are analysed with the aim of calculating *traffic statistics.* Such type of analysis can help identify the most accessed pages and content, and may therefore highlight user preferences. These preferences may not have been previously anticipated during the design stage, and may therefore need to be incorporated by restructuring the application's hypertext structure.

Traffic analysis is not able to detect users' navigational behaviour. To allow a deeper insight into users' navigation paths, the research community has investigated techniques to reconstruct user navigation from log

files [17,19,22,23]. Most techniques are based on the extensions of Web logging mechanisms, used to record additional semantic information about the content presented in the pages accessed. This information can later be used to understand observed frequent paths and corresponding pages [6]. Such extensions exploit Semantic Web techniques, such as RDF annotations for mapping URLs into a set of ontological entities. Also, recent work [23,56] has proposed conceptual enrichment of Web logs through the integration of information about a page's content and the hypertext structure deriving from the application's conceptual specification. The reconstruction of a user navigation can then be incorporated into automatic tools, which provide designers and evaluators with statistics about identified navigation paths. Such paths can be useful to evaluate and improve an application's organisation with respect to its actual usage.

User navigation paths can also be analysed by means of *Web usage mining* techniques, which apply data mining techniques on Web logs to identify associations between visited pages and content [17,22]. With respect to the simple reconstruction of user navigation, Web usage mining can discover unexpected user behaviour, not foreseen by the application designers. The user behaviour can be a symptom of a poor design, rather than a defect. The aim is to identify possible improvements that accommodate such user needs.

Different techniques can be used to mine Web logs. Mining of *association rules* is probably the one used the most. Association rules [1] are implications of the form $X \Rightarrow Y$, stating that in a given session where the `X` log element (e.g. a page) is found, the `Y` log element is also very likely to be found. Methods for discovering association rules can also be extended to the problem of discovering *sequential patterns*. These are extensions of association rules to the case where the relation between rule items specifies a temporal pattern. The sequential pattern of the form `X.html` $\Rightarrow$ `Y.html` states that users, who in a session visit page `X.html`, are also likely to next visit page `Y.html` in the same session [62].

The discovery of association rules and sequential patterns is interesting from the Web usage perspective, because the results produced can provide evidence of content or pages that are frequently associated. If this behaviour is not supported by proper navigational structures, connecting such content to pages, then it can suggest possible improvements to ease content browsing.

A drawback of Web usage mining techniques is that they require a substantial amount of pre-processing[2] [17,61]. In particular, user session identification can be very demanding, since requests for pages tracked in

[2] To extract user navigation sessions containing consistent information, and to format data in a way suitable for analysis.

Web logs may be compromised due to proxy servers, which do not allow the unique identification of users [18]. Solutions to circumvent this problem are illustrated in [18].

Comparison of Methods

User testing provides reliable evaluations, because its results are based on user samples representative of the population of real users. It helps evaluators overcome problems, such as lack of precision of predictive models whenever the application domain is not supported by a strong and detailed theory. User testing, however, has a number of drawbacks. The first is the difficulty to select a sample representative of the population of real users, since the identification of such a population is sometimes not straightforward. A sample that does not represent the correct population provides results unlikely to be of use. The second drawback is that it can be difficult to train users, within a limited amount of time, to master advanced features of a Web application. This can lead to shallow conclusions, in most cases only related to the simple application features. The third drawback is that the limited amount of time available for user tests makes it difficult to mimic real usage scenarios. Such scenarios require the provision of a real environment where the application is to be used, and also the motivations and the goals that users may have in real-life situations [37]. Failure to reproduce such a context may lead to unrealistic results and conclusions. Finally, the fourth drawback is that user observation provides little information about the cause of a problem, since it deals primarily with the symptoms [21]. Not understanding the underlying cause has implications for an application's redesign. In fact, the new design can remove the original symptoms, but if the underlying cause remains, a different symptom may result.

Unlike user testing, inspection methods enable the identification of the underlying cause of a problem. Inspectors know exactly which part of the design violates a usability principle, and how. The main advantage of inspection methods, compared to user testing, is that they can be carried out with a smaller number of people, i.e., they are conducted by usability and human factor experts, who can detect problems and possible future faults of a complex system in a limited amount of time. This is in our view a relevant point, which strongly supports the use of usability evaluations during the design activities. In fact, it constitutes an inexpensive add-on to existing development practices, easily enabling the integration of usability goals into those of the software design and development [21]. Furthermore, inspection techniques can be used early on in the development process lifecycle, using if necessary design specifications, whenever a prototype is not yet available.

The three main disadvantages of inspection methods are, first, the great subjectivity of the evaluation – different inspectors may produce incomparable outcomes; and second, the strong dependency upon inspectors' skills. Third, experts can misjudge the reactions of real users in two ways, i.e. not detecting potential problems, or discovering problems that will not be relevant for real users.

According to Brooks [12], usability inspection methods cannot replace user testing because they are not able to analyse aspects, such as trade-offs, the entire interface acceptability, or the accuracy of a user's mental model. Also, they are not suitable to define the most usable interface out of several, or anything that relates to a preference. However, usability testing cannot predict if an interface will "just do the job" or will "delight the user"; this type of information is, however, important within the context of a competitive user market share. Therefore it may be beneficial also to consider features that can distinguish an interface from good to excellent, rather than to focus solely on its problems, which is what usability inspection does.

The analysis of Web server logs seems to solve a series of problems in usability evaluation, as it may reduce the need for usability testing. Also, with respect to the experimental settings, it offers the possibility of analysing the behaviour of a higher number of users, compared to user tests, increasing the number of attributes that can be measured, and the reliability of the detected errors. However, the use of Web server log files is not without problems of its own. The most severe relates to the meaning of the information collected and how much it describes real users' behaviour. Even when logs are effective in finding patterns in the users' navigation sessions, they cannot be used to infer users' goals and expectations, central for a usability evaluation.

5.5 Automatic Tools To Support Evaluations

Automatic tools have been suggested as the most efficient means to treat repetitive evaluation tasks, without requiring much time and skills from human resources. There are three main categories of Web evaluation tools [11], which cover a large set of tests for usability and accessibility:

- *Tools for accessibility analysis*. Measures that can be automatically collected by these tools correspond to official accessibility criteria (such as those prescribed by W3C), and refer to properties of the HTML page coding, such as browser compatibility, use of safe colours, appropriate colour contrast, etc. Examples are Bobby [10], A-Prompt [3], and LIFT [36].

- *Tools for usability analysis.* These tools verify usability guidelines by analysing an application's design. They operate predominantly at the presentation layer, with the aim of discovering problems, such as the consistency of content presentation and navigation commands (e.g. link labels, colour consistency). They often neglect structural and navigation problems, although recent proposals (see for example [23]) plan to address such issues, by focusing on the identification of structural problems in the hypertext definition. Examples are CWW [9], WebTango [31], and WebCriteria SiteProfile [65].
- *Tools for Web usage analysis.* These tools allow the computation of statistics about an application's activities, and mine data about user behaviour. The majority of commercial tools (see for example [2,4]) are traffic analysers. Their functionality is limited to producing the following reports and statistics [22]:
  - *Site traffic reports*, such as total number of visits, average number of hits, average view time.
  - *Diagnostic statistics*, such as server errors and pages not found.
  - *Referrer statistics*, such as search engines accessing the application.
  - *User statistics*, such as top geographical regions.
  - *Client statistics*, such as users Web browsers and operating systems.

Research has been recently proposed to analyse user navigation paths, and to mine Web usage [7,17,42].

While the adoption of automatic tools for Web log analysis is mandatory, an important observation must be made about the first two categories of tools. Such tools constitute valuable support to reduce the effort required to manually analyse an entire application with respect to all of the possible usability problems. However, they are not able to exhaustively verify usability issues. In particular, they cannot assess any properties that require judgement by a human specialist (e.g. usage of natural and concise language). Also, automatic tools cannot provide answers about the nature of a discovered problem and the design revision that can solve it.

Automatic tools are therefore useful when their use complements the activity of human specialists, since they can execute repetitive evaluation tasks to inspect the application, and highlight critical features that should later be inspected by evaluators.

5.6 Evaluation of the DEI Application

The DEI application has been developed by means of an iterative development process, in which several incremental application versions have

been released, evaluated, and improved based upon problems raised by the evaluations. Such a process has been enabled by the ease of prototype generation, due to the adoption of a modelling language, WebML [14], and its accompanying development tool [13,66], offering a visual environment for composing WebML-based specifications of an application's content and hypertext, and a solid XML and Java-based technology for automatic code generation.

The guidelines introduced in Sect. 5.3 have been taken into account during the application design. However, in order to further validate usability, several evaluation sessions, through different evaluation methods, have been conducted. In particular:

- Inspection sessions to examine the hypertext specification have been conducted, using an automatic tool aimed at discovering structural problems related to the definition of erroneous or inconsistent navigation mechanisms.
- After the application delivery, Web logs have been analysed to verify if the application structure envisioned by the application designers matched user needs, or if some unexpected behaviours could occur.
- The released prototypes, as well as the delivered final application, have been analysed through heuristic evaluations, to further identify problems that could not easily be revealed through the analysis of design specifications.

5.6.1 Design Inspection

Design inspections have been carried out over 14 successive versions of the DEI application, by applying different procedures to evaluate structural properties, such as its internal consistency and the soundness of navigation mechanisms.

Thanks to the availability of the XML-based representation of the hypertext specification, generated by the adopted development tool, the inspection was conducted automatically through the adoption of WQA (Web Quality Analyzer) [23], an XSL-based tool able to parse the XML specification for retrieving usability problems. In particular, the tool inspects the application design, looking for configurations that are considered potential sources of problem. Thus, it executes analysis procedures aimed at verifying if any configurations found violate usability.

In the following section we will illustrate two main problems we identified within the content management area used by Web administrators.

Consistency of Operation Design

Some of our inspection procedures aimed to verify the design consistency of content management operations, used to create and modify an application's content, within the content management area. In particular, they had to identify all occurrences of operations within pages, and to verify if their invocation, and the visualisation of results after their execution, was coherent across the entire application.

Fig. 5.4 plots the history of the Modify Termination (MT) evaluation procedure along several releases of the DEI application. Such procedure allowed us to evaluate and measure the consistency of visualisation results for content modification operations, with respect to two possible variants: visualisation of the modified content *(i)* in the same page where the operation was invoked (*Same Page Termination* variant), or *(ii)* in a new page (*Different Page Termination* variant). The procedure thus entailed:

1. To identify all the modification operations specified in the application's hypertext;
2. To compute the statistical variance (a value between 0 and 1) with respect to the occurrences of the two different termination variants, normalised with respect to the best-case variance (see [24] for further details).

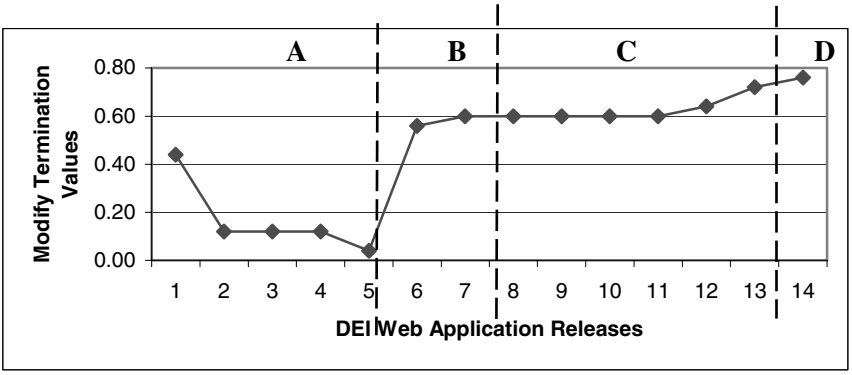

Fig. 5.4. History of the MT computation along the different DEI application releases

The plot in Fig. 5.4 highlights four different phases (from A to D) in the application's development. Phase A is characterised by a limited care regarding the design consistency. The initial high value of the computed measures for release 1 depended on the limited number of modification operations in the application at that time. However, as soon as the number of modification operations in the following releases started to grow, the

consistency value for the termination of modification operations decreased, and reached its lowest value (0.04) in release 5. At this point, the evaluators raised the problem. Therefore during phase B, the application designer modified the application, trying to use a single design variant (the Same Page termination variant) in almost every case. Release 6 clearly shows the improvement obtained by the re-engineering activity, with the variance value going from 0.04 to 0.54. Improvement is also noted in relation to the percentages of use of the two variants in releases 5 and 6, as detailed in Table 5.3.

Table 5.3. The percentage of the occurrences of the two different Modify pattern variants within releases 5 and 6

	Different page termination	Same page termination
Release 5	42,86%	57,14%
Release 6	12,5%	87,5%

Starting from release 7 (phase C), the MT measure computation has reached a constant value – no modifications have been applied to the modification operation. From release 12 to 14 (phase D), we have instead assisted with improving the application's consistency. The last computed value for the MT metrics was 0.76, which corresponds to an acceptable level of consistency with respect to the set usability goals.

Identification of Dead-ends

Besides verifying consistency, some inspection tasks were performed to discover structural weaknesses within the application's hypertext. One particular inspection procedure, executed on DEI's hypertext specification, aimed to discover *dead-ends*. Dead-ends are pages reachable by different navigation paths preventing the user from navigating further. The only choice they give to a user is to go back (e.g. by hitting a browser's "back" button). These pages either have no outgoing links, or activate operations that end up where they started, thus making navigation difficult.

While analysing the entire application's structure we identified 20 occurrences of the *dead-end* pattern. A closer look at each occurrence revealed that all *dead-ends* were pages reached whenever a database update operation failed. In such a situation, the user is presented with a page displaying an error message, and is unable to navigate further, to recover from the error. According to the "ease of browsing" usability criterion, the designer should have inserted a link to allow a user to go back to the initial page from which the operation had been invoked.

It is worth noting that dead-end pages would have been difficult to find by means of user testing. The reason is that the investigator would need to induce database-related errors in order to obtain one of these pages. This is therefore an example where the analysis of design specifications (automatic or not) to verify structural properties can support the identification of usability problems that may be difficult to find.

5.6.2 Web Usage Analysis

Once the WEI application was deployed, we carried out a Web usage analysis. To do so, we analysed Web logs to reconstruct user navigation patterns, and to identify possible critical situations that had been encountered by users. In particular, the analysis focused on navigational access mechanisms.

Navigation Sequences for Accessing a DEI Member's Page

The computation of traffic statistics was used to identify that, apart from the application's Home Page, the other most visited URL corresponded to the page showing the details for a single DEI member, with links to the member's publications, personal home page, and research areas. The navigational facilities provided by the application enabled us to examine the means employed by users to reach the page. As can be observed in Fig. 5.2, the page is reachable through a navigational access that takes a user from the Home Page to an index page, thus providing two different ways to reach a single member's page:

- The first using an index of member categories (e.g. professor, lecturer), which allows for the selection of a category and, using a different page, the selection of a specific member from an alphabetically ordered list.
- The second using a search-based direct access.

Given these navigational access facilities, we wanted to monitor their usage, to find out whether they showed any usability problems. In order to identify navigational sequences, we adopted a module of the WQA tool [23], which is able to analyse Web logs and to reconstruct user navigation. The analysis of logs for 15 days showed that the navigational sequence from the index page to the DEI member page was followed about 20,000 times during that period, and that:

- The indexes of categories and members were used less than 900 times.
- Times users went through the search box more than 19,000.

These results suggested either that users did not know which category to use when looking for a DEI member, or that the navigational access was not easy to use and needed improvements. This feedback is currently being taken into account by the application designers, to assess the merits of re-designing the application.

Another problem related to the access to the DEI member pages was identified while carrying out a mining session on the DEI Web logs, to discover possible association rules [42]. The mining query was aimed at discovering the page sequences most frequently visited. Results showed that requests for a "research area" page were later followed by a request to a DEI member page. A possible reason for this behaviour could be that users perceive the "research area" page as an access page for the DEI members. This result supports the view that users are not making use of the navigational access on DEI members, as envisioned by the application designers.

Navigation Sequences for Accessing a DEI Member's Publications

Another problem also identified was related to accessing the DEI member's publication details. The "Publication Details" page[3] can be reached from four distinct pages: "Search Publications,"[4] "All Publications,"[5] "DEI Member Publications"[6], and "Project Publications"[7]. Yet again, our goal was to discover the mostly used path to reach the "Publication Details" page. To do so, we gathered data on all the navigation sequences that contained the "Publication details" page.

The analysis of 15 days of Web logs revealed that the "Publication Details" page had been visited 440 times during that period, organised as follows:

- The "Publication Details" page was reached 420 times from the "DEI Member Publications" page.
- Of the remaining 20 times: the page "Publication Details" was reached 8 times from "Project Publications", 7 times from the "All Publications" page, and twice from the "Search Publications" page.

---

[3] Page that shows the full details of a publication.
[4] Page that provides a keyword-based search.
[5] Page that displays the list of all the publications.
[6] Page that shows the publication list for a specific DEI member.
[7] Page that displays the publications produced in the context of a specific research project.

To reach the "Publication Details" page the "DEI Member Publications" page seems very likely to occur, therefore the results were not surprising. However, the small number of times that other pages were used to reach the "Publication Details" page was a concern. To understand these results, we inspected the application's hypertext design, to consider all the navigational sequences that reached the "Publication Details" page from pages other than the "DEI Member Publications" page. The inspection results showed that the "All Publications" and "Search Publications" pages were only reachable through links displayed in the "Publication Details" page. Therefore the reason for the low usage of such pages is that they were not visible to users.

Note that a problem such as this could not be identified solely by analysing the design, as this suggests that both pages can be reached using links from "Publication Details" page. The analysis of the design would not therefore take both pages as "unreachable". In addition, this problem could not be identified using a heuristic evaluation as the hypertext structure employed does not violate any usability principles. This is therefore an example that supports the need for observing real users' behaviour.

Heuristic Evaluation of Hypertext Interface

To achieve more complete and reliable evaluation results, design inspection and Web usage analysis were complemented with a heuristic evaluation session, conducted by expert evaluators from outside the design team. The aim of this evaluation was to assess the usability of the hypertext's presentation layer, which had not been addressed by the two previous evaluations.

Nielsen's heuristics were considered as the benchmark criteria. Results indicated problems related to the effectiveness of the language adopted. For example, the DEI Home Page (see Fig. 5.2) shows content categories that are related to the application core concepts. The same content categories are presented within each page using a permanent navigation bar. However, the category "Staff" in the Home Page is displayed as "People" in the navigation bar available in all remaining pages. The solution to the naming inconsistency is always to use the same category name.

Another problem, also related to naming conventions, is related to the inconsistent semantics of the "Details", within a page that shows the publications for a DEI professor (see Fig. 5.5). The link "Details" on the left side of the navigation bar is used as a link to the DEI "Member" page. However, the link "Details" underneath each publication is used as a link to the detailed description of a particular publication. Interpreting the problem in the light of Nielsen's heuristics, we can therefore observe that:

1. A system-oriented language has been employed, rather than a user-orientated language. In fact, "Details" was the term constantly referred to by application designers, during the application's development, to represent the presentation of an information entity's detailed contents (e.g. DEI "Member" and "Publication"). This problem is therefore an example where the interface has not been user-centred. Such a problem can be solved by assigning meaningful names to links clearly indicating the contents to be displayed in the target page.
2. To adopt the same name to denote two different concepts means users have to "remember" the interaction model implemented, rather than allowing them to "recall" such a model. The interface does not make objects, actions, and options visible, thus requiring the user to remember how to reach the content across different application areas, and different interaction sessions.

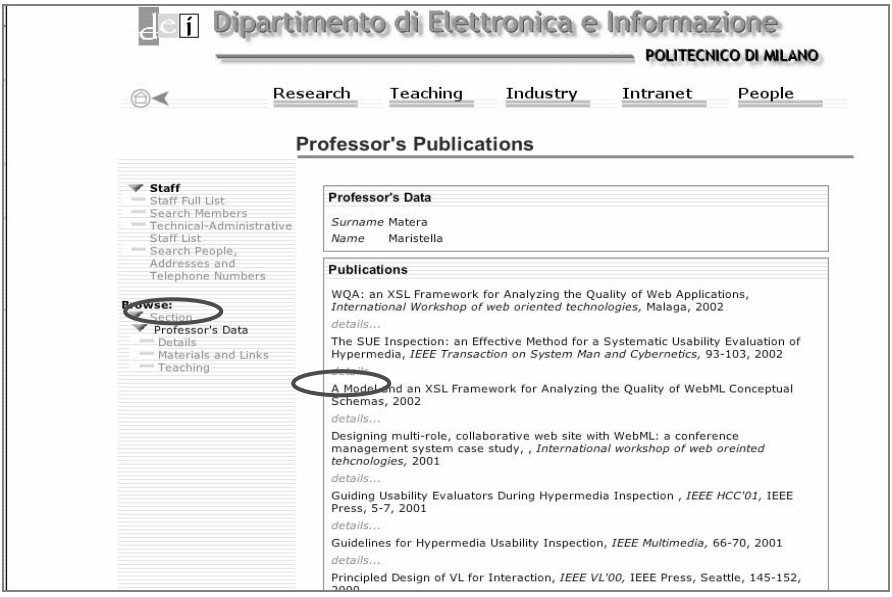

Fig. 5.5. Ambiguous semantics for the "Details" link name

5.7 Concluding Remarks

Web applications are quickly growing in number and complexity, becoming the *de facto* standard for distributed applications that require human interaction. The increasing number of Web users, the diversity of application domains, content, the complexity of hypertext structures and

interfaces, all encourage the use and measurement of usability as a determinant factor for the success of such applications.

The process by which engineering principles are applied to developing Web applications started only recently [5,14,57]. Web engineering provides application designers with a collection of tools and languages to accelerate the development process, and to enforce a level of syntactic correctness, allowing for semi or complete, automatic code generation. Syntactic correctness prevents a designer from specifying an application that has defects. However, a quality application is more than a piece of defect-free code.

Applications that incorporate usability engineering into their development process are expected to comply with quality requirements. In particular [38]:

1. Evaluation is the key for assuring quality: the effort devoted to an evaluation can directly determine the quality of the final application.
2. To be effective, an evaluation must rely upon suitable and validated usability criteria.

This chapter provided an overview of methods currently adopted in assessing the usability of Web applications, and criteria that can be applied to the evaluation of Web applications. In addition, this chapter also highlighted the advantages and drawbacks of different usability methods so as to help practitioners choose the most suitable method with respect to the evaluation goals.

Independent of the method chosen, practitioners and researchers suggest that a sound usability evaluation plan should include the use of different, complementary methods, to ensure the completeness of the evaluation results. The characteristics of each method determine their effectiveness in discovering a specific class of usability problems.

The adoption of automatic tools can improve the reliability of the evaluation process. As reported in [11], tools for automatic analysis can address some of the issues that prevent developers from adopting evaluation methods. In particular, tools are systematic, fast, and reliable, and can be effectively adopted for tackling repetitive and time-consuming evaluation tasks. Also, tools may allow developers to code and execute procedures for the verification of in-house guidelines, making them easily enforceable.

However, tools may help verify structural properties, but fail to assess properties that require specialised human judgement, to provide answers explaining the nature of a given problem, and suggestions on how to fix it. Automatic tools are therefore very useful when their use complements the activity of human specialists.

References

1 Agrawal R, Imielinski T, Swami A. 1993) Mining Association Rules Between Sets of Items in Large Databases. In: Proceedings of *ACM-SIGMOD 93*, Washington, DC, May, pp 207–216

2 Analog. (2005) http://www.analog.cx. (accessed on 18th January 2005)

3 A-Prompt Project. (2005) http://aprompt.snow.utoronto.ca/ (accessed 18 January 2005)

4 AWSD-WebLog. (2005) http://awsd.com/scripts/weblog/index.shtml (accessed 18 January 2005)

5 Baresi L, Garzotto F, Paolini P (2001) Extending UML for Modeling Web Applications. In: Proceedings of the 34th Annual Hawaii International Conference on System Sciences, Maui, USA, January

6 Berendt B, Hotho A, Stumme G (2002) Towards Semantic Web Mining. In: Proceedings of the 1st International Semantic Web Conference, Sardinia, Italy, June. Springer, Berlin, LNCC. 2342, pp 264–278

7 Berendt B, Spiliopoulou M (2000) Analysis of Navigation Behaviour in Web Sites Integrating Multiple Information Systems. J Very Large Data Bases, 9(1):56–75

8 Bias RG, Mayhew DJ (1994) Cost-justifying usability. Academic Press, Boston, MA

9 Blackmon MH, Polson PG, Kitajima M, Lewis C (2002) Cognitive Walkthrough for the Web. In: Proceedings of the 2002 International Conference on Human Factors in Computing Systems, Minneapolis, USA, April, pp 463–470

10 Bobby. (2005) http://bobby.watchfire.com/bobby/html/en/index.jsp (accessed 18 January 2005)

11 Brajnik G (2004) Using Automatic Tools in Accessibility and Usability Assurance. In: Proceedings of the 8th International Workshop on User Interface for All, Vienna. June, Springer, Berlin, LNCC 3196, pp 219–234

12 Brooks P (1994) Adding Value to Usability Testing. In: Nielsen J, Mack RL (eds) Usability Inspection Methods. Wiley, New York, pp 255–271

13 Ceri S, Fraternali, P (2003) Architectural Issues and Solutions in the Development of Data-Intensive Web Applications. In: Proceedings of the First Biennial Conference on Innovative Data Systems Research, Asilomar, USA, January

14 Ceri S, Fraternali P, Bongio A, Brambilla M, Comai S, Matera M (2003) Designing Data-Intensive Web Applications, Morgan Kaufmann, San Francisco, CA

15 Ceri S, Fraternali P, Matera M (2002) Conceptual Modeling of Data-Intensive Web Applications. IEEE Internet Computing, 6(4):20–30

16 Conallen J (2002) Building Web Applications with UML, Addison-Wesley, Boston, MA

17 Cooley R (2003) The Use of Web Structures and Content to Identify Subjectively Interesting Web Usage Patterns. ACM Transactions on Internet Technology 3(2):93–116

18 Cooley R, Mobasher B, Srivastava J (1999) Data Preparation for Mining World Wide Web Browsing Patterns. J Knowledge and Information Systems, 1(1):5–32

19 Cooley R, Tan P, Srivastava J (2000) Discovery of Interesting Usage Patterns from Web Data. In: Proceedings of the 1999 International Workshop on Web Usage Analysis and User Profiling, San Diego, USA, August. Springer, Berlin, LNCC 1836, pp 163–182

20 Dix A, Finlay J, Abowd G, Beale R (1998) Human-Computer Interaction, 2nd edn. Prentice Hall, London

21 Doubleday A, Ryan M, Springett M, Sutcliffe A (1997) A Comparison of Usability Techniques for Evaluating Design. In: Proceedings of the 1999 Symposium on Designing Interactive Systems: Processes, Practices, Methods and Techniques, Amsterdam, the Netherlands, August, pp 101–110

22 Eirinaki M, Vazirgiannis M (2003) Web Mining for Web Personalization. J ACM Transactions on Internet Technology, 3(1):1–27

23 Fraternali P, Lanzi PL, Matera M, Maurino A (2004) A Model-Driven Web Usage Analysis for the Evaluation of Web Application Quality. Web Engineering, 3(2):124–152

24 Fraternali P, Matera M, Maurino A (2002) WQA: An XSL Framework for Analyzing the Quality of Web Applications. In: Proceedings of the Second International Workshop on Web-Oriented Software Technologies, Malaga, Spain, June

25 Garzotto F, Matera M (1997) A Systematic Method for Hypermedia Usability Inspection. New Review of Hypermedia and Multimedia, 6(3):39–65

26 Hull L (2004) Accessibility: It's Not Just for Disabilities Any More. ACM Interactions, 41(2):36–41

27 Hutchins EL, Hollan JD, Norman DA (1985) Direct manipulation interfaces. Human-Computer Interaction, 1:311–338

28 IBM (2005) Ease of Use guidelines. http://www-306.ibm.com/ibm/easy/eou_ext.nsf/publish/558 (2005). (accessed 18 January 2005)

29 ISO (1997) ISO 9241: Ergonomics Requirements for Office Work with Visual Display Terminal (VDT) Parts 1–17

30 Ivory MY, Hearst MA (2001) The State of the Art in Automating Usability Evaluation of User Interfaces. ACM Computing Surveys, 33(4):470–516

31 Ivory MY, Sinha RR, Hearst MA (2001) Empirically Validated Web Page Design Metrics. In: Proceedings of the ACM International Conference on Human Factors in Computing Systems, Seattle, USA, April, pp 53–60

32 Jeffries R, Desurvire HW (1992) Usability Testing vs. Heuristic Evaluation: Was There a Context? ACM SIGCHI Bulletin, 24(4):39–41

33 Jeffries R, Miller J, Wharton C, Uyeda KM (1991) User Interface Evaluation in the Real Word: A Comparison of Four Techniques. In: Proceedings of the ACM International Conference on Human Factors in Computing Systems, New Orleans, USA, pp 119–124

34 Kantner L, Rosenbaum S (1997) Usability Studies of WWW Sites: Heuristic Evaluation vs. Laboratory Testing. In: Proceedings of the ACM 1997 International Conference on Computer Documentation, Snowbird, USA, pp 153–160

35 Lewis JR (1995) IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instruction for Use. Human-Computer Interaction, 7(1):57–78

36 LIFT. (2005) http://www.usablenet.com (accessed 18 January 2005)

37 Lim KH, Benbasat I, Todd PA (1996) An Experimental Investigation of the Interactive Effects of Interface Style, Instructions, and Task Familiarity on User Performance. ACM Transactions on Computer-Human Interaction, 3(1):1–37

38 Lowe D (2003) Emerging knowledge in Web Development. In Aurum A, Jeffery R, Wohlin C, Handzic M (eds) Managing Software Engineering Knowledge. Springer, Berlin, pp 157–175

39 Lynch P, Horton S (2001) Web Style Guide: Basic Design Principles for Creating Web Sites, 2nd edn. Yale University Press, New Heaven, CT

40 Madsen KH (1999) Special Issue on The Diversity of Usability Practices. J Communications of the ACM, 42(5)

41 Marchionini G, Shneiderman B (1988) Finding Facts vs. Browsing Knowledge in Hypertext Systems. IEEE Computer, 21(1):70–80

42 Meo R, Lanzi PL, Matera M, Esposito R (2004) Integrating Web Conceptual Modeling and Web Usage Mining. In: Proceedings of the 2002 International ACM Workshop on Web Mining and Web Usage Analysis, Seattle, USA, August

43 Molich R, Nielsen J (1990) Improving a Human-Computer Dialogue. Communications of the ACM, 33(3):338–348

44 Nielsen J (1992) The Usability Engineering Lifecycle. J IEEE Computer, 25(3):12–22

45 Nielsen J (1993) Usability Engineering. Academic Press, Cambridge, MA

46 Nielsen J (1994) Special Issue on Usability Laboratories. Behavior and Information Technology, 13(1)

47 Nielsen J (1994) Guerrilla HCI: Using Discount Usability Engineering to Penetrate Intimidation Barrier. In: Proceedings of the Cost-Justifying Usability, Academic Press, Cambridge, MA

48 Nielsen J (1995) Multimedia and Hypertext Internet and Beyond, Academic Press, London

49 Nielsen J (2000) Web Usability. New Riders, New York

50 Nielsen J, Landauer TK (1993) A Mathematical Model of the Finding of Usability Problems. In: Proceedings of the ACM 1993 International Conference on Human Factors in Computing Systems, Amsterdam, Netherlands, April, pp 296–213

51 Nielsen J, Mack RL (1994) Usability Inspection Methods. Wiley, New York

52 Nielsen J, Molich R (1990) Heuristic Evaluation of User Interfaces. In: Proceedings of the ACM 1990 International Conference on Human Factors in Computing Systems, Seattle, USA, April, pp 249–256

53 Norman DA (1991) Cognitive Artifacts. In: Proceedings of the Designing Interaction: Psychology at the Human–Computer Interface. Cambridge University. New York, pp. 17–38

54 Polson P, Lewis C, Rieman J, Wharton C (1992) Cognitive Walkthrough: A Method for Theory-based Evaluation of User Interfaces. Man-Machine Studies, 36:741–773

55 Preece J, Rogers Y, Sharp H, Benyon D, Holland S, Carey T (1994) Human-Computer Interaction. Addison-Wesley, New York

56 Punin JR, Krishnamoorthy MS, Zaki MJ (2002) LOGML: Log Markup Language for Web Usage Mining. In: Proceedings of the Third International Workshop on Web Mining and Web Usage Analysis, San Francisco, USA, August, pp 88–112

57 Schwabe D, Rossi G (1998) An Object Oriented Approach to Web-Based Applications Design. Theory and Practice of Object Ssystems, 4(4):207–225

58 Shneiderman B (1992) Designing the User Interface. Strategies for Effective Human-Computer Interaction. Addison-Wesley, New York

59 Shneiderman B (2000) Universal Usability. Communications of the ACM, 43(5):84–91

60 Shneiderman B, Byrd D, Croft WB (1998) Sorting out searching. Communications of the ACM, 41(4):95–98

61 Srivastava J, Cooley R, Deshpande M, Tan PN (2000) Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. ACM Special Interest Group on Knowledge Discovery in Data Explorations, 1(2):12.23

62 Stroulia E, Niu N, El-Ramly M (2002) Understanding Web Usage for Dynamic Web-site Adaptation: A Case Study. In: Proceedings of the 4th International Workshop on Web Site Evolution, Montreal, Canada, October, pp 53–64

63 Theofanos MF, Redish J (2003) Bridging the gap between accessibility and usability. ACM Interactions, 10(6):36–51

64 Virzi RA (1992) Refining the Test Phase of Usability Evaluation: How Many Subjects is Enough? Human Factors, 34(4):457–468

65 WebCriteria SiteProfile. (2005) http://www.coremetrics.com (accessed 18 January 2005)

66 WebRatio Site Development Studio. (2005) http://www.webratio.com (accessed 18 January 2005)

67 Wharton C, Rieman J, Lewis C, Polson P (1994) The Cognitive Walkthrough Method: A Practitioner's Guide. In: Nielsen J, Mack RL (eds) Usability Inspection Methods, Wiley, New York, pp 105–140

68 Whiteside J, Bennet J, Holtzblatt K (1988) Usability Engineering: Our Experience and Evolution. In: Helander M (ed.) Handbook of Human-Computer Interaction. Elsevier, Amsterdam pp 791–817

69 Wilson TD (2000) Human Information Behavior. Informing Science, 3(2):49–55

70 Wurman RS (1997) Information Architects, Watson-Guptill, New York

71 W3C Consortium – Extended log file format. (2005) http://www.w3.org/TR/WD-logfile.html. (accessed 18 January 2005)

72 W3C Consortium - WAI-Web Content Accessibility Guidelines 2.0. (2005) W3C-WAI Working Draft. http://www.w3.org/TR/WCAG20/ (accessed 18 January 2005)

Authors' Biographies

Maristella Matera is Assistant Professor at Politecnico di Milano, where she teaches Databases and Computer Science Fundamentals. Her research interests focus on design methods and tools for Web applications, and in particular concentrate on conceptual modelling quality, Web log mining, personalisation of Web applications, context-aware Web applications, multimodal Web interfaces, Web application usability and accessibility. She is author of about 50 papers on the previous topics and of the book Designing Data-Intensive Web Applications, published by Morgan Kaufmann in December 2002. She has served as co-chair for several editions of the "Web Technologies and Applications" track at ACM SAC, and of the CAISE Workshop UMICS (Ubiquitous Mobile Information and Collaboration Systems). She is also regularly a member of the programme committee of several conferences and workshops in the field of Web Engineering.

A more detailed curriculum vitae and list of publications can be found at: http://www.elet.polimi.it/people/matera.

Francesca Rizzo is a junior researcher at Politecnico di Milano, where she is a lecturer for the Human Computer Interaction Laboratory. She obtained her PhD in Telematics and Information Society from the University of Siena in 2003. In the last five years she has taught human computer interaction and interaction design at the University of Siena and at Politecnico di Milano. Her fields of interest are human computer interaction (HCI), user centered design (UCD), usability evaluation and activities analysis. She has worked in many European research projects. Currently, her research is focused on Web application usability and accessibility, e-learning, story telling technologies design and evaluation for children. She is author of about 20 papers on the previous topics. She has served as reviewer for

several conferences and journals such as ICWE (International Conference on Web Engineering) and JWE (Journal of Web Engineering).

Giovanni Toffetti Carughi graduated in Information Engineering at Politecnico di Milano in 2001. His thesis work focused on the extension through plugins of the WebML methodology for designing and automatically generating data-intensive Web applications. He worked for three years in the software industry both as a developer for WebRatio and as analyst and consultant in different industrial Web applications such as Acer-Euro portals, ABI-Pattichiari, Nortel-Consip, MetalC.

He is currently a PhD student at Politecnico di Milano and his topics of interest are conceptual modelling, model transformation, Web application engineering, Web log analysis, human computer interaction, rich internet applications, and image compression.