Web Usability: Principles and Evaluation Methods

Chapter · March 2006

DOI: 10.1007/3-540-28218-1_5

3 authors, including:

Maristella Matera
Politecnico di Milano
266 PUBLICATIONS **4,275** CITATIONS

SEE PROFILE

Francesca Rizzo
Politecnico di Milano
71 PUBLICATIONS **1,275** CITATIONS

SEE PROFILE

Web Usability: Principles and Evaluation Methods

Maristella Matera, Francesca Rizzo, Giovanni Toffetti Carughi

Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza Leonardo da Vinci, 32 – 20133 – Milano – Italy

Abstract: Current Web applications are very complex and high sophisticated software products, whose usability can heavily determine their success or failure. Defining methods for ensuring usability is one of the current goals of the Web Engineering research. Also, much attention on usability is currently paid by Industry, which is recognizing the importance of adopting methods for usability evaluation before and after the 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 introduced concepts and methods.

Keywords: Web usability, Evaluation Methods, Web Usability Principles, Development Process.

1 Introduction

The World Wide Web has having a significant impact on the access to the huge quantity of information available through the Internet. Web-based applications have influenced several domains, providing access to information and services by a variety of users showing different characteristics and backgrounds. Users visit Web sites, and also return back to previously accessed sites, if they easily find useful information, organized in a way that facilitates access and navigation, and presented

according to a well-structured layout. In few words, the acceptability of Web applications by users strictly relies on their *usability*.

Usability is one relevant factor of the quality of Web applications. Recently, it has been receiving great attention, being recognized as a fundamental property for the success of Web applications. Defining methods for ensuring usability is therefore one of the current goals of the Web Engineering research. Also, much attention on usability is currently paid by Industry, which is recognizing the importance of adopting usability methods during the development process, for verifying the usability of Web applications before and after their deployment. Some studies have in fact demonstrated how the use of such methods enables cost saving, with a high cost-benefit ratio, since they reduce the need for changes after the application delivery [40, 50].

1.1 Usability in the Software Lifecycle

Traditional software engineering lifecycles do not explicitly address usability. They suggest different activities, from the initial inception of an idea until the product deployment, among which testing is conducted at the end of the cycle with the aim of checking if the application design satisfies the high-level requirements agreed with the customers, and if it is complete and internally consistent. In order to achieve usable applications, it is however necessary to augment the standard lifecycle for addressing explicitly usability issues. This objective does not imply simply adding some activities; rather it requires involving appropriate techniques which span the entire lifecycle [20].

Given the emergent need for usability, during last years traditional development processes have been extended for enabling the fulfilment of usability requirements. Evaluation methods have been adopted at any stage of the process for verifying the usability of incremental design artefacts, as well as of the final product. This has resulted into the proposal of the so-called *iterative design* [58, 16] for promoting usability throughout the whole product lifecycle.

With respect to more traditional approaches suggesting a top-down, analytic approach (as for example the waterfall model), iterative design recognises the development process be complemented by a bottom-up, synthetic approach, in which the requirements, the design and the product gradually evolve, becoming step by step well defined. The essence of iterative design is that the only way to be sure about the effectiveness of some design decisions is to build and evaluate them, through the use of application prototypes. The design can be then modified, to correct any

false assumption detected during the evaluation activities, or to accommodate new emerged requirements; the cycle of design, evaluation, and redesign, must be repeated as often as necessary.

In this context, *usability evaluation* is intended 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 in this model: it is relevant at all the stages in the lifecycle, not only at the end of the product development. All the 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 emphasizes the role of prototyping and evaluation, the discovery of new requirements, and the importance of involving diverse stakeholders– including users. But what makes iterative development more than merely well-intentioned trial-and-error? Usability engineering became the banner under which diverse methodological endeavours were carried out through the 1990s:

- It proposes that iterative development has to be managed according to explicit and measurable objectives, called ‘‘usability specifications’’, to be identified early during the development process. Explicit usability goals are therefore incorporated within the design process, emphasising that the less expensive way for obtaining usable products is to consider usability issues early in the lifecycle, so as to reduce the need of modifying the design at the end of the process [44, 45].
- It claims for "discount usability engineering", which consists in the adoption of easy to apply, but still efficient, evaluation techniques, so as to encourage developers to consider usability issues throughout the whole development cycle [47].

1.2 Chapter Organization

The aim of this chapter is to illustrate a set 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 specialization for the Web, we will illustrate some usability criteria that support Web usability in two manners: on one hand, they can guide the design process, providing guidelines about how to organize the application by means of usable solutions; on the other hand, they drive the evaluation process, providing benchmarks for usability verification. We will then illustrate evaluation

methods to be tackled during the whole development process – both during design and after the application deployment - that are based on the intervention of usability specialists, or on the involvement of real users.

In order to exemplify some of the introduced concepts, we will illustrate some remarkable issues in the design and evaluation of a real Web application, the institutional site of 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 site. It consists of:

- A *public area*, publishing information about the Department staff, and their teaching and research activities. It receives about 9.000 page requests per days 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 front end for creating or updating content to be published via the Web application.

2 Defining Web Usability

Usability is generally acknowledged as a factor of system quality representing the answer to many frustrating interactions with technology. It describes the quality of products and systems from the point of view of humans who use them.

Different definitions of usability have been so far 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 to 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 the Nielsen's definition, usability refers to:

- *Learnability*: the ease of learning the functionality and the behavior 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 to easy recover.
- *User's satisfaction*: the measure in which the user finds the system pleasant to use.

The previous principles can be further specialized and decomposed into finer-grained criteria that can be verified through different evaluation methods. The resulting advantage is that more precise and measurable criteria contributes towards setting an engineering discipline, where usability is not just argued, but is systematically approached, evaluated and improved [44, 45].

When applying usability to Web applications, some refinements need to be operated over 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 discovering new ones 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 be therefore considered as the ability of Web applications to support such tasks with effectiveness, efficiency and satisfaction. Also, the above mentioned Nielsen's usability principles can be interpreted as follows [48]:

- *Web application learnability* must be interpreted as the ease for Web users to understand from Home Page 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 so as contents are easy to understand and navigational mechanisms are easy to identify.

- *Web applications efficiency* means that users that want to find some contents can reach them quickly through the available links. Also, when users get to a page, they must be able to orient themselves and understand the meaning of the page with respect to their navigation starting point.
- *Memorability* implies that, after a period of non-use, users are still able to get oriented within the hypertext, for example by means of navigation bars pointing to landmark pages.
- *Few errors* mean that in case users have erroneously followed a link, they should be able to return to their previous location.
- *Users' satisfaction* finally refers to the situation in which users feel that they are in control with respect to the hypertext, thanks to the comprehension of available contents and navigational commands.

In order to be evaluated, the previous criteria can be further refined into more objective and measurable criteria. Section 3 will introduce a set of operational criteria for Web application design and evaluation.

2.1 Usability and Accessibility

Recently, the concept of usability has been extended to cover *accessibility*. Accessibility focuses on those 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 unpaired users. A number of works in literature also assign to this concept a broader meaning, defining it as the ability of an application to support any users identifying, retrieving and navigating its contents [26, 63]. In fact, any user can take advantage of Web applications providing accessible contents, especially in all those circumstances when the Web access becomes difficult due to specific contexts of use, such as adopting voice-based devices (e.g., cellular phones) while moving by car. According to this meaning, accessibility can be therefore considered as 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 as 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 focusing on those usability criteria enhancing the effectiveness and efficiency of navigation and orientation mechanisms. The second goal can be achieved by operating on the page mark-up, and in particular:

- *Separating presentation from content and navigation design*, which enables presenting the same contents and navigational commands according to multiple presentation modalities suitable for different devices.
- *Augmenting multimedia content with textual descriptions*, so as it can be presented even through alternative browsing technologies, such as screen readers assisting unpaired 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 or black and white screens, and when pointing devices are not available.

WCAG recommendations provide 14 guidelines, each one featuring criteria specifying how it can be applied in a specific context. For more details the reader is referred to [72].

3 Web Usability Criteria

According to the Usability Engineering approach, a cost-effective way for increasing usability is to address it since the early phases of the application development. A solution for achieving this goal is to take into account some criteria that refine general usability principles (such as those presented in Section 2), suggesting how the application must be organized to conform to usability requirements [45]. On one hand, 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]. On the other hand, they constitute the background for the evaluation activity.

Table 1. The ten Nielsen's 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 recognize, 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

As established by several methods recently introduced in Web Engineering [5, 14, 57], the development of Web applications must focus on three separate dimensions, i.e., data, hypertext and presentation design, each dimension being accompanied by a number of criteria. 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 organizing the presentation layer of Web applications [39, 49]. Table 1 for example summarizes 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 peculiar requirements, conventions and constraints characterizing the design of contents and hypertexts in Web applications. This section therefore proposes a set of criteria that suggest how Web applications should be organized, both at data and hypertext level, for supporting information finding and browsing, and user orientation, which represent three fundamental aspects having great impact over the 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 the adoption of few high-level conceptual abstractions for systematically conceiving the overall structure of the application, avoiding dwelling on implementation and mark-up coding of single pages, and especially focusing on the organization "in-the-large" of the information content and the hypertext structure. In particular, the criteria are based on the assumption that, as also suggested by a consolidated recommendation coming from the field of Human Computer Interaction and Human Factor studies [41, 69, 70], the retrieval and fruition of contents by end users is significantly affected by the way in which the content itself is conceived and designed, and then delivered through the hypertext interface. The usability of Web applications thus requires the complete understanding and the accurate modeling of data resources. As such, differently from previous proposals [25, 48, 49], our criteria are organized so as general principles are expanded into two sets of more practical guidelines, one set suggesting how to structure content, the other one proposing the definition of usable navigation and orientation mechanisms for content access and browsing.

3.1 Content Visibility

In order to understand the structure of the information space offered by the application, and get 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 contents are identified and adequately structured. To fulfill this requirement, the application design should start from the identification of the information entities modeling 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 centered on such contents, and will gradually evolve by detailing their structure in terms of elementary components, and adding further auxiliary contents for accessing and browsing them.

Hypertext Modularity

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

- The hypertext can be organized in *areas,* i.e., modularization constructs grouping pages that publish homogeneous contents. Each one should refer to a given core concept identified at 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 (for example, 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 enhancing 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 contents and the location within the hypertext where they are placed. Once learned, they also support orientation and error recovering, because 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 organization of the information content, the hypertext of the public Web site is organized into four areas,

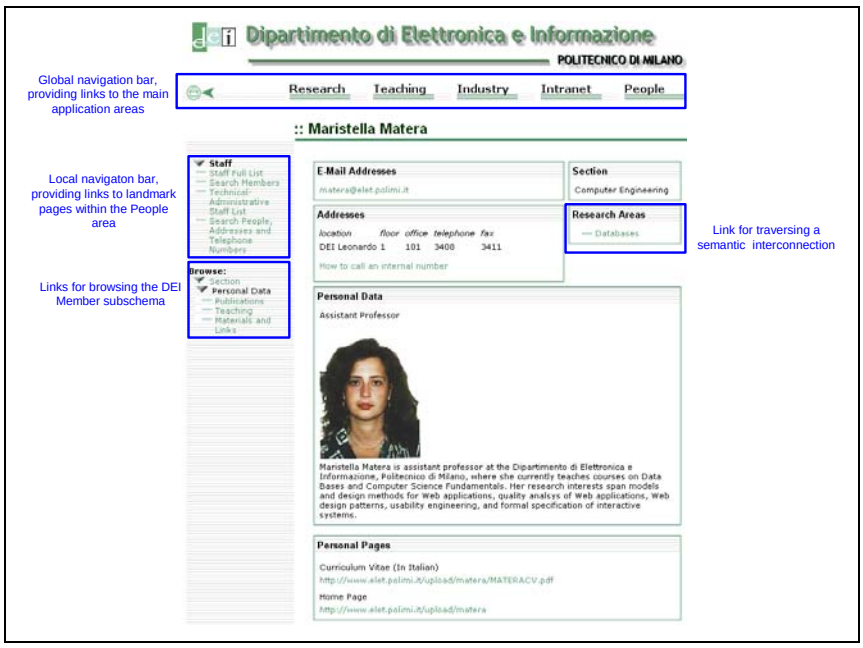

Fig. 1. Page organization, with global and local landmarks, and core, peripheral/, and interconnection sections

Research, *Teaching*, *Industry*, and *People*, each one corresponding to a single core concept.

Each page within the application therefore 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. Fig. 1 for example illustrates a page from the People area, which displays some 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 area *People* are placed in the top region of the left-side bar.

3.2 Ease of Content Access

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

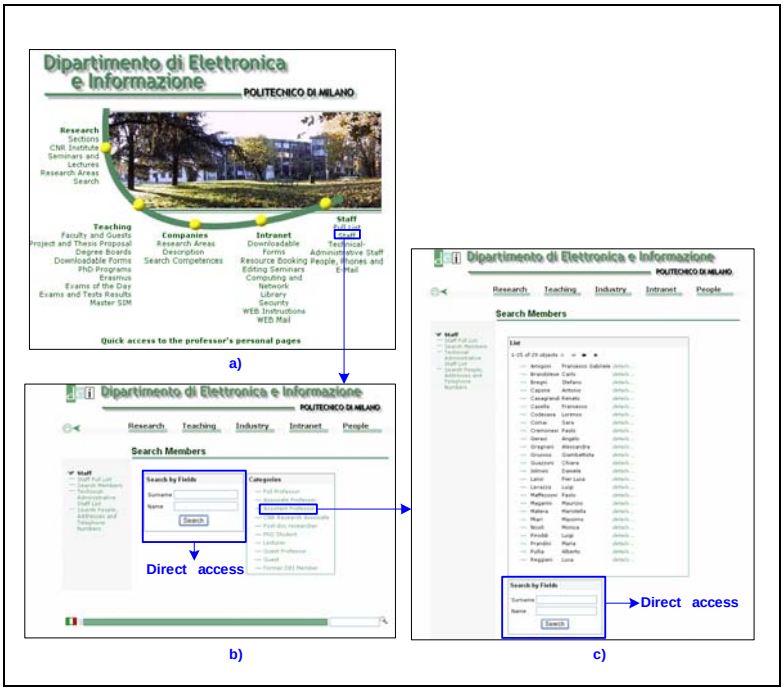

Fig. 2. Hierarchy of indexes in the navigational access to the DEI Member, consisting of the Home Page (a), the Member Categories page (b), and the Category Member Index page (c). Pages (b) and (c) also include a keyword-based search for direct access

Identification of Access Information Concepts

The design of access paths for retrieving core content items can be facilitated if designers augment the application contents with *access concepts* corresponding to classification criteria or contexts over core concepts, enabling users to progressively move 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 correspond to every core concept.

Navigational Access and Search-Based Access

In order to facilitate the access to specific instances of core concepts, access concepts defined at data level should be used to construct *navigational access mechanisms* that typically consist of multi-level indexes, possibly distributed on several *access pages*, bridging pages with

a high visibility, such as 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 very often complemented with *direct access*, i.e., keyword-based search mechanisms, which let users bypass navigation and rapidly reach the desired information objects. Direct access mechanisms are essential ingredients in interfaces (such as those of mobile devices) that are unable to support multiple navigation steps. In traditional hypertext interfaces, they enhance orientation when users get lost by moving along navigational access mechanisms [60, 49].

Pairing navigational and direct access with explicit visibility about available categorisations and free text queries, and a regular use of these access mechanisms within the hypertext greatly enhances content accessibility.

Content Access in the DEI Application

In the DEI application, each area is provided with both navigational and direct access. Fig. 2 represents the contextual access path defined for the core concept *DEI Member*. It consists of a hierarchy of indexes, developed through different access pages, which lets users move from some broadest people categories, presented in the application Home Page (e.g., academic staff) to pages listing narrower sub-categories (e.g., all the categories of the academic staff), then to the list of members in a selected sub-category, from which users can select one person's name and access the corresponding page. Each page also provides direct access, by means of a keyword-based search for directly reaching single DEI members.

3.3 Ease of Content Browsing

> *Users must be able to easily identify possible auxiliary contents related to each single core concept, as well as the available interconnections among different core concepts.*

Core Concepts Structuring and Interconnection

The ease of use and the learnability of a Web application can be enhanced by supporting the users understanding of the content structuring and of the semantic interconnections defined among different content classes. Therefore, when the identified core concepts represent a structured and

complex concept, it is recommended to expand them – via top-down design - into a composite data structure, collectively representing the whole core concept, and characterized by:
– A *central information content* - which expresses the concept's main content and provides an individual identity to each individual core concept.
– Some other *peripheral information elements*, which complete the concept description.

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

Organization of Core Pages

In order to highlight the structure of each core concept, and the interconnections holding among different concepts, pages devoted to core concept presentation should contain at least three sections:
– A *core section* should clearly convey contents associated with the core concept.
– A *peripheral* section should highlight auxiliary information – if any – completing the core concept.
– An *interconnection* section should represent 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 role of information in the page. If systematically repeated through the application, it enhances consistency among the components displayed by pages [49], and therefore becomes a key of interpretation of the hypertext supporting a conscious focus shift by users. Also, if explicitly annotated on the page mark-up code, it is the key for building intelligent page readers enabling accessibility by any users.

Content Browsing in the DEI Application

As an example of content browsing organization in the DEI application, let us consider the DEI Member page represented in Fig. 1. The page features as core section a central region that presents the attributes qualifying a DEI

Member (e-mail address, postal address, department section, bio, list of personal pages). The page then includes three groups of links:

– One group, referring to the page's peripheral section, point to pages publishing more details about each DEI member, such as the list of publications and courses.
– Other links, representing the page's interconnection section, enable a semantic interconnection, pointing to the research area the member belongs to.

Such a page structure also applies to pages in the other application areas.

4 Evaluation Methods

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

The main goals of evaluation are to assess the application functionality, to verify the effect of its interface on the user, and to identify any specific problem with the application, such as aspects which show unexpected effects when used in their intended context [20]. Evaluating Web applications in particular consists in verifying if the application design allows users to easily retrieve and browse contents, and invoke available services and operations. This therefore implies not only having appropriate contents and services available into the application, but also making them easily reachable by users through appropriate hypertexts.

Depending on the phase in which evaluation is performed, it is possible to distinguish between *formative evaluation*, which takes place during design, and *summative evaluation*, which takes place after the product has been developed, or even when any prototype version is ready. During the early design stages the goal of the formative evaluation is to check the design team understanding of the users' requirements, and to test design choices quickly and informally, thus providing feedback to the design activities. Later on, the summative evaluation can support the detection of users' difficulties, and the improvement and the upgrading of the product.

Within these two broad categories, there are different methods that can be used at different stages of the product development. The most

commonly adopted are *user testing*, where the real users are studied, and *usability inspection*, which is conducted by specialists. Recently, *Web usage analysis* has also emerged as a method for studying user behaviours through the computation of access statistics and the reconstruction of user navigation on the basis of Web access logs.

The rest of this section is devoted to illustrate the main features of these three classes of evaluation methods, also highlighting their advantages and drawbacks.

4.1 User Testing

User testing deals with real behaviours, observed from some representative of real users [46]. It requires that users perform a set of tasks through physical artefacts, being them prototypes or systems, while the experimenter observes users behaviours and collects empirical data about the way users execute the assigned tasks [20, 55, 68]. Typical data collected during user testing are user execution time, number of errors, and user satisfaction. After the test completion, the collected data are then interpreted and used to ameliorate the level of the application usability.

Usability testing is explicitly devoted to analyse in details how users interact with the application for accomplishing well-defined tasks. This feature determines the difference between usability testing and beta testing, largely applied in Industry. Beta testing is conducted on the final product: after the application release the end users are contacted and interviewed about their satisfaction. Conversely, usability testing is conducted 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 together with specific redesign suggestions.

In order to avoid any inconvenient related to the reliability of results, the design of the test and its execution have to be carefully planned and managed. A good usability testing could be therefore articulated as follows:

1. **Defining the goals of the test**. The objectives of the evaluation can be generic, as for example the improvement of end users satisfaction and the design of a product easy to use; or they can be specific, as for example the evaluation of the effectiveness of a navigational bar for user orientation; the readability of labels, etc.
2. **Defining the sample of users that will participate in the test**. The sample of subjects for the test has to be representative of the entire end users population. Possible criteria that can be used to define the test

sample are: the user experience (experts vs. novices); the age; the frequency of use of the application; the experience with similar applications. The number of the participants can vary, depending on the objectives of the test. Nielsen and Molich [52] affirm that the 50% of the most important usability problems can be identified with three users. Other authors claim that the involvement of 5 users enables discovering the 90% of the usability problems [47, 64].

3. **Selecting tasks and scenarios**. The tasks submitted to users during the test have to be real; in other words they have to represent the activities that people would perform on the application. Scenarios can be selected from the results obtained during requirements elicitations; differently, they can be purposely prepared to test unexpected situations.
4. **Establishing how to measure the level of usability of the system**. Before conducting a usability testing, it is necessary to define the parameters that will be used to measure the results. The type of measures can vary from the subjective ones, as the user satisfaction, the difficulty of use, etc., to the most objective and quantitative ones, as the task completion time, the number and the typology of errors, the number of successfully accomplished tasks, the number of time users invoke help (verbal, on line help, manual). The results will be anonym and the participants will be informed about their use. Besides observation, the experimenter may use other techniques for gathering data about task execution: the *think aloud*, in which the subject is required to tell explicitly all the actions s/he is trying to tackle, the reason of her/his actions, 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 experimenter stimulates participants to reflect on the events of the test session. It is worth noting that such techniques do not provide ways for collecting data about users' satisfaction. Such a kind of subjective measure can be instead obtained through *survey techniques*, based on the use of questionnaires and interviews [35, 58], to be submitted to users after the execution of the testing tasks.
5. **Preparing the needed material and the experimental environment**. The experimental environment has to be organized equipping it with a computer and a video camera for recording user activities, establishing the roles of the experimental team members, and preparing any supporting material (manuals, pencils and papers, etc.). Executing the test in a laboratory is not mandatory. Prior to the testing session, a pilot trial is necessary for checking and possibly refining all the test procedures.

4.2 Inspection Methods

It is recommendable to conduct user testing for studying users while using prototypes. However, this activity is quite expensive. Also, in the development process the feedback has to start coming in at earlier stages, preferably before even the first software or hardware has been built. The previous reasons have lead to the definition of usability inspection methods, to be used by developers to predict usability problems that could be detected through users testing.

Usability inspection refers to a set of evaluation techniques that are an evolution from prior function and code inspection methods used in Software Engineering for debugging and improving code. According to such methods evaluators examine usability related aspects of an application, trying to detect violations of established usability principles [51], and then provide feedback to designers about possible design improvements. The inspectors can be usability specialists, or also designers and engineers with special expertise (e.g., knowledge of specific domains or standards). In any case, the application of such methods relies on a good understanding of the usability principles, and more specifically of how they apply to the specific application to be analysed, and on the particular ability of the evaluators in discovering critical situations where principle violations occur.

Usability inspection methods have been proposed when the issue of cost effectiveness started guiding methodological work on usability evaluation [8]. The cost of user studies and laboratory experiments became a central issue. Therefore, many proposals were made for usability evaluation techniques 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]. Among them, the most commonly used are *heuristic evaluation* [45, 51], in which usability specialists judge whether the application properties conform to established usability principles, and *cognitive walkthrough* [54, 67], which uses detailed procedures for simulating users' problem-solving processes, trying to see if the functions provided by the application are efficient for users, and lead them to the next correct actions. In the sequel of this section we will describe such two methods. For a detailed description of other inspection techniques the reader is referred to [51].

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

Fig. 3. An example of table for reporting heuristic violations

Heuristic Evaluation

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

During the evaluation session, each evaluator goes individually through the system interface at least twice. The first step is to get a feel of the flow of the interaction and the general scope of the application; the second is to focus 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 Fig. 3 for an example). Reporting problems in relation to heuristics enables the easy generation of a revised design, 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.

Heuristic evaluation is especially valuable when time and resources are short, because skilled evaluators, without needing the involvement of representative users, can produce high quality results in a limited amount of time [34]. In principle, heuristic evaluation can be conducted by only one evaluator. However, in an analysis of six studies, it has been assessed that single evaluators are able to find only the 35% of the total number of the existent usability problems [43], and that different evaluators tend to find different problems. Therefore, the more experts are involved in the evaluation, the more problems it is possible to find. Fig. 4 shows the percentage of usability problems found by having different numbers of evaluators, as reflected by a mathematical model defined in [50]. From the curve it is evident that reasonable results can be obtained by having only five evaluators, and certainly not less than three.

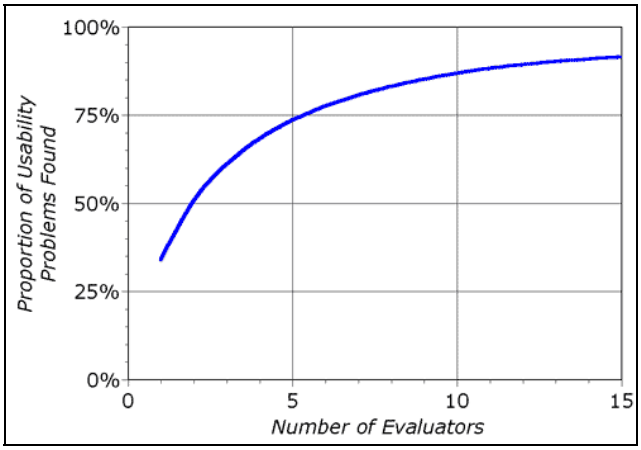

Fig. 4. The percentage of usability problems found by heuristic evaluation when using different numbers of evaluators [50]

As already mentioned for inspection methods in general, heuristic evaluation has a number of drawbacks. As highlighted in [21, 33, 34], its major drawback is its high dependence upon the skills and the experiences of the evaluators. Nielsen states that novice evaluators with no usability expertise are poor evaluators, usability experts are 1.8 times as good, and application domain and usability experts (the double experts) are 2.7 as good [44, 45]. This means that the specific experience with the specific category of applications really improves the evaluators' performance.

Cognitive Walkthrough

In cognitive walkthrough, the user's problem solving process - i.e., what the users will do in specific situations of use and why - is simulated [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 the users would accomplish trying to reach the objectives defined in the scenario, by means of the identification of the relations occurring among user goals, user actions, and the visible states of the application interface [27]. As such, cognitive walkthrough is particularly suitable for the detection of problems affecting the application learnability.

Cognitive walkthrough is largely applied to evaluate presentation aspects in the application interface. Its use is recommended in the advanced phases of the Web application development, for evaluating 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 on the interface by an expert evaluator. The evaluator executes such tasks, and after the completion of each elementary action s/he tries to interpret the application answer, and to evaluate the steps forward for the achievement of the end user 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 out alternative goals?

During this interpretation process, it is also possible that the user/evaluator needs to change its initial goal because it is impossible to reach it. 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 is communicated back to the design team.

4.3 Web Usage Analysis

A relatively new direction in the evaluation of Web applications deals with Web usage analysis [30], performed on the record of user accesses to the application pages, collected in a Web server log [61] according to one of the available standard formats [71]. After Web applications are deployed, Web usage analysis can be employed to analyze how users exploit and browse the information provided by the Web site. For instance, it can help discovering those navigation patterns which correspond to high Web usage, or those which correspond to early leaving.

Very often, Web logs are analyzed with the aim of calculating *traffic statistics.* Such a type of analysis can help identify the most accessed pages and contents, and may therefore highlight some user preferences, not

detected at design time, that might need to be accommodated by restructuring the hypertext.

Traffic analysis is not able to detect users' navigational behavior. To allow deeper insight into users' navigation paths, the research community has been studying techniques to reconstruct user navigation from log files [17, 19, 22, 23]. Most of them are based on extensions of Web logging mechanisms, for recording additional semantic information about contents displayed in the accessed pages, to make sense of the observed frequent paths and of pages on these paths [6]. Such extensions exploit Semantic Web techniques, such as RDF annotations for mapping URLs into a set of ontological entities. Also, some recent works [23, 56] have proposed conceptual enrichment of Web logs through the integration of information about the page content and the hypertext structure deriving from the application conceptual specifications. The reconstruction of user navigation can be then incorporated into automatic tools providing designers and evaluators with statistics about the identified navigation paths that can be useful for evaluating and improving the application organization with respect to the actual application usage.

User navigation paths can also be analyzed by means of *Web Usage Mining* techniques, which consist in applying data mining techniques over Web logs for identifying interesting associations among visited pages and contents [17, 22]. With respect to the simple reconstruction of user navigation, Web Usage Mining can discover unexpected user behaviors, not foreseen by the application designers, which can be the symptom of design lacks, not necessarily errors. The aim is to identify possible amendments for accommodating such user needs.

Different techniques can be used to mine Web logs. Mining of association rules is probably the one most used. 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, also the Y log element is 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 among 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 $X.html$, afterwards in the same session are also likely to visit page $Y.html$ [62].

The discovery of association rules and sequential patterns is interesting from the Web usage perspective, because the results produced can evidence contents or pages that frequently appears in association. If the discovered behaviour is not supported by appropriate navigational

structures connecting such contents and pages, then it can suggest possible changes for improving the ease of content browsing.

A drawback of Web usage mining techniques is that they require a substantial amount of pre-processing [17, 61] to clean the logs, extract user navigation sessions containing consistent information, and formatting data in a way suitable for analysis. In particular, user session identification can be very demanding [18]. Requests for pages tracked into the Web logs must be grouped in order to identify the navigation paths of single users; but this phase may suffer for problems mainly due to proxy servers, which do not allow the unique identification of users, generally based on IP address. Some solutions for circumvent this problem are illustrated in [18].

4.4 Comparison of Methods

User testing provides reliable evaluations, because it involves samples of real users. It allows evaluators to overcome the lack of precision manifested by predictive models when the application domain is not supported by a strong and detailed theory. Such a method, however, has a number of drawbacks. It is difficult to select a proper sample of the user community: an incorrect sample may lead to wrong perceptions about the user needs and preferences. It is difficult, in a limited amount of time, to train users to master the most sophisticated and advanced features of a Web application; not well trained users may produce "superficial" conclusions, only related to the most immediate features of the application. Furthermore, it is difficult, in a limited amount of time, to reproduce actual situations of usage, which requires setting up the environment where the application is going 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 "artificial" conclusions rather than to realistic results. Finally, user observation gives little information about the cause of the problem, because it primarily deals with the symptoms [21]. Not understanding the underlying cause has implication for re-design. In fact, the new design can remove the original symptom, but if the underlying cause remains, a different symptom can be triggered.

Differently from user testing, inspection methods enable the identification of the underlying cause of the problem, because the inspectors exactly report on which part of the design violates usability principles and how. Their main advantage, with respect to user testing, is that they involve fewer experienced people, i.e., experts of usability and human factors, which can detect problems and possible future faults of a complex system in a limited amount of time. This is a relevant point,

which strongly supports the viability and the acceptability of usability evaluation during the design activities. In fact, it constitutes a low expensive addition to existing practices, so enabling the easy integration of usability goals with the goals of the efficient software design and development [21]. Furthermore, inspection techniques can be used very early in the development process, even when prototypes are not available and evaluation must be conducted over design specifications.

The main disadvantages of inspection methods are however the great subjectivity of the evaluation - different inspectors may produce not comparable outcomes - and the heavy dependence upon the inspector skills. Also, experts can misjudge the reactions of real users in two ways, i.e., not detecting potential problems, or figuring out problems that will not be significant for users.

According to Brooks [12] usability inspection methods cannot replace user testing because they are not able to analyze aspects such as trade-offs, or the whole acceptability of the interface, or the accuracy of the mental model of the user. Also, they are not suitable for defining the most usable interface out of several, or things that are a matter of preference. On the other hand, usability testing cannot tell whether the interface will "just do the job or delight the user"; this kind of information is, however, important in the competition for users on the market. As an attitude matter, the design of an interface might therefore benefit from somebody looking at what is good and successful, not always focusing on the problems, as usability inspection does.

Web server logs analysis seems to solve a series of problems in the field of the usability evaluation, since it might reduce the need for usability testing involving sample of real users. Also, with respect to experimental settings, it offers the possibility of analyzing the behavior of a high number of users, thus increasing the number of evaluated variables and the reliability of the detected errors. However log files are not without problems. The most severe one is about the meaning of the information collected and how much it describes the real people behavior. In fact, even when they are effective in finding patterns in the users' navigation sessions, such techniques do not solve the problem of how to infer users' goals and expectations, which are central information for usability evaluation.

5 Automatic Tools Supporting Evaluation

Even though the cost (in terms of time and effort) of inspection methods and user testing is not particularly high, and their effectiveness has been so far largely proved for augmenting the quality of Web applications, very often it happens that evaluation is not systematically performed on the whole application and at each development step. The reasons can be related to a number of factors [11], such as the need of shortening the application release time, the availability of incomplete and vague design specifications, or more simply the lack of resources for conducting evaluation. Also, inspection and user testing are often difficult to manage when a large number of people are involved in the evaluation process. Claims about the number of evaluators that are enough to detect usability breakdown, as well as the nature of the qualitative data per se - that does not allow conducting systematic verifications and comparison - are therefore pulling for automatic tools able to efficiently treat the most repetitive evaluation tasks, without requiring much time and skills by 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*, like Bobby [10], A-Prompt [3], LIFT [36], etc. The metrics implemented 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.
- *Tools for usability analysis*, such as CWW [9], WebTango [31], WebCriteria SiteProfile [65], that analyse site design for verifying usability guidelines. They mostly operate at the presentation layer, with the aim of discovering problems such as the consistency for presentation of contents and navigation commands (e.g., link labels, colour consistency, etc.). Very often, they neglect structural and navigation problems. Some recent proposals (see for example [23]) are now trying to address such issues, by focusing more on the identification of structural lacks in the hypertext definition.
- *Tools for Web usage analysis*, which allow calculating statistics about site activities, and mining data about user behavior. The majority of the commercial tools (see for example [2, 4]) are traffic analyzers. As also described in [22], their functionality is limited to producing:
  - *Site traffic reports*, such as total number of visits, average number of hits, average view time, etc.
  - *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 user's Web browsers and operating systems.

Some research works have been recently proposed for the analysis of user navigation paths and for Web usage mining [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 a valuable support for reducing the efforts required to evaluators for analysing "by hand" the whole application with respect to all the possible usability lacks. However, they are not able to verify exhaustively usability issues. In particular, they cannot assess all those properties that require judgements by human specialists (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 very useful when their use complements the activity of human specialists, since they can execute repetitive evaluation tasks for inspecting the application and highlighting critical features that are worth to be later inspected by evaluators.

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 with respect to the problems raised by evaluation. Such a process has been enabled by the ease of prototype generation, due to the adoption of a modeling language, WebML [14], and its accompanying development tool [13, 66], offering a visual environment for composing WebML-based specifications of the application content and hypertext, and a solid XML and JAVA-based technology for automatic code generation.

The guidelines introduced in Section 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:

- Some inspection sessions have been conducted over the hypertext specification by means of an automatic tool, with the aim of discovering structural problems related to the definition of erroneous or inconsistent navigation mechanisms.

- After the application delivery, Web logs have been analyzed for verifying if the application structure envisioned by application designers matched well user need, or if some unexpected behaviors occur.
- The released prototypes, as well as the delivered final application have been analyzed through heuristic evaluation, for identifying further problems that could not be easily revealed through the analysis of design specifications.

6.1 Design Inspection

Design inspection has been conducted 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 has been 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 some configurations that are considered potential sources of problems. Thus, it executes some analysis procedures aimed at verifying whether found configurations violate usability.

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

Consistency of Operation Design

Some inspection procedures concentrated on verifying the consistency of the design of content management operations for creating and modifying application contents within the content management area. In particular, this evaluation activity consisted in identifying all the occurrences of operations within pages, and verifying if their invocation, and the visualization of results after their execution were coherent across the whole application.

Fig. 5 for example plots the history of the Modify Termination (MT) evaluation procedure along several releases of the DEI applications. Such a procedure has allowed us to evaluate and measure the consistency of result visualization for content modify operations, with respect to two possible variants: visualizing the modified content *(i)* in the same page where the

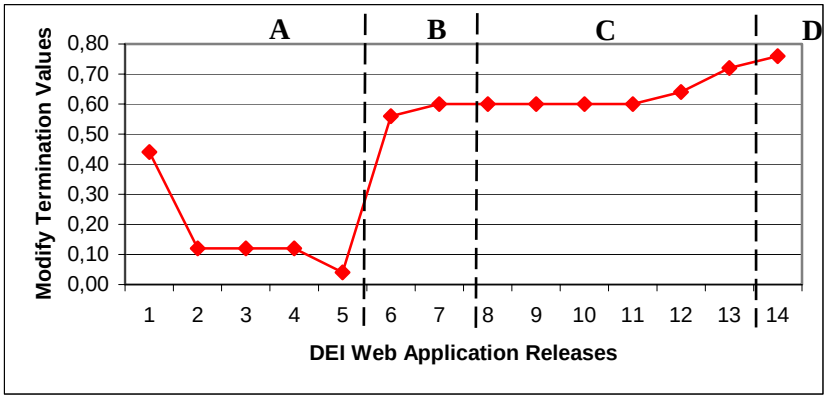

Fig. 5. History of the MT computation along the different DEI application releases

operation is invoked (*Same Page Termination* variant), or *(ii)* in a new page (*Different Page Termination* variant). The procedure thus consisted of:

1. Identifying all the modify operations specified in the application hypertext;
2. Computing the statistical variance (a value between 0 and 1) with respect to the occurrences of the two different termination variants, normalized with respect to the best-case variance (see [24] for major details).

The graphic in Fig. 5 highlights four different phases (from A to D) in the application development. Phase A is characterized by a scarce care about design consistency. The initial high value of the computed metrics for release 1 only depends on the limited number of modify operations featured at that time within the application. However, as soon as the number of modify operations started growing in the following releases, the consistency value for the termination of modify operations decreased, until reaching 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 one only design variant (the Same Page termination variant) in almost every case. The release 6 already reflects this re-engineering, featuring an improvement of the variance value, from 0.04 to 0.54. The improvement can also be noted from the percentages of use of the two variants in release 5 and 6, as detailed in Table 2.

Table 2. 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 metric computation has produced a constant value - no modifications have been applied on the modify operation. From release 12 to 14 (phase D), we have instead assisted to the always-increasing effort of designers to improve consistency. The last computed value for the MT metrics is 0.76 – which corresponds to an acceptable level of consistency with respect to the considered usability goals.

Identification of Dead-Ends

Besides verifying consistency, some inspection tasks have been conducted for discovering structural weaknesses within the application hypertext. At this proposal one interesting inspection procedure executed over the DEI hypertext specification was aimed at discovering the so called *dead-ends*. Dead-Ends are pages reachable by different navigation paths preventing the user to navigate further: the only choice the final user has is hitting the "back" button of the browser. These pages either have no outgoing links or activate operations ending up where they started, thus making navigation difficult for users.

By performing the application schema analysis we have identified twenty occurrences of the above pattern throughout the whole application. Giving a closer look to each occurrence, we have noticed that all the dead-ends in the application were pages reached whenever a database update operation fails. In this situation, the user therefore gets in a page displaying a message error, without any possibility to navigate further and recover from the occurred error. According to the "ease of browsing" usability criteria, in this case the designer would insert a link to allow user to go back to the initial page from which the operation has been invoked.

It is worth noting that dead-end pages would have been difficult to spot through user testing. The reason is that the experimenter would need to induce some database related errors in order to be forwarded to one of these pages. This is therefore an example of situation where the analysis of design specifications (automatic or not) aimed at verifying structural properties can support the identification of subtle usability problems.

6.2 Web Usage Analysis

Once the application has been deployed, we have analysed Web logs for reconstructing user navigation and identifying possible critical situations encountered by users. In particular, the analysis has focused on navigational access mechanisms.

Navigation Sequences for Accessing a DEI Member's Page

By calculating some traffic statistics, we noticed that, apart from the home page, the most visited URL corresponded to the page showing the details of a single DEI member, and providing links to consult the member's publications, personal home pages, and research areas (see Fig. 1). Being this page the most accessed one, we wanted therefore to monitor in which way users reached it, given the navigational facilities provided by the application.

As can be observed in Fig. 2, the page is reachable through a navigational access that conducts user from the Home Page to an index page providing two different ways to reach a single member:

1. An index of member categories, which allows one to select a category and then, in a different page, a specific category member from an alphabetically ordered list.
2. 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 analyze Web logs and reconstruct user navigation. Analysing 15 days of logs, we found that the navigational sequence from the index page to the DEI member page was followed about 20,000 times, and that:

1. Less than 900 times the indexes of categories and members were used.
2. More than 19,000 times users went through the search box.

We came therefore to the conclusion that users probably would not know which category to use when looking for a DEI member or that in general the navigational access is not easy to use and needs some improvements. Such a design amendment is currently under evaluation by the application designers for an application re-design.

A further problem about the access defined over the DEI member pages came out by executing a mining session over the DEI Web logs for detecting interesting association rules [42]. The mining query was aimed at

discovering the page sequences most frequently visited. Results showed that requests of a research area page where later followed by 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; thus also in this case they do not exploit the navigational access over DEI members as envisioned by designers.

Navigation Sequences for Accessing a DEI Member's Publications

Another discovered problem regards the access defined for the DEI member's publication details. The "Publication Details" page, showing the full details of a publication, can be reached from pages "Search Publications", providing a keyword-based search, "All Publications", displaying the list of all the publications, "DEI Member Publications", showing the publication list for a specific DEI Member, and "Project Publications", publishing the publications produced in the context of a specific research project. Once again, we wanted to focus on which is the most used path for reaching publication details, and therefore we collected all navigation sequences composed of any page plus "Publication details".

Analyzing 15 days of Web logs, we discovered that page "Publication Details" was visited about 440 times. Among them:

- 420 times users came from the "DEI Member Publications" page.
- Of the remaining 20 times: 8 were from "Project Publications", 7 from "All publications", and just 2 from "Search Publications".

Apart from the fact that browsing from a researcher page to his/her publications can be considered common, the incredibly low number of other accesses rang a bell. We then inspected the hypertext design, to consider all the navigational sequences that reached the publication details without passing through the "DEI Member Publications" page. The result showed clearly that pages "All publications" and "Search publications" are only reachable through links displayed in the "Publication Details" page. Therefore the reason of the low usage of such pages is that they were not visible to users.

It is worth noting that such a problem could not be identified simply analysing the design, since from specifications it results that the two pages are reachable through some links (those coming from page "Publication details"). The analysis of design would not therefore mark the two pages as "unreachable". Also, the problem could not be identified through heuristic evaluation, because apparently such a hypertext configuration does not

violate any usability principle. This is therefore a typical example that can emerge from the observation of the real user behaviour.

6.3 Heuristic Evaluation of the Hypertext Interface

In order to achieve more complete and reliable evaluation results, design inspection and Web usage analysis have been complemented with a heuristic evaluation session, conducted by expert evaluators from outside the design team. The aim of this further evaluation session has been studying the usability of the presentation layer of the hypertext interface. Such issues were not addressed through the previous evaluations.

Nielsen heuristics have been considered as benchmark criteria for the evaluation. Some problems related to the effectiveness of the adopted language have been detected. For example, as can be noted in Fig. 2, the DEI Home Page shows content categories that are related to the application core concepts. The same content categories are presented within each page in the form of permanent navigation bar. However, the category “Staff” in the Home Page is displayed as “People” in the persistent navigation bar displayed in all the other pages. This naming inconsistency should be solved by adopting the same category name in every place where the category must be displayed.

Another problem about naming conventions is caused by the inconsistent semantics assigned to the link “Details” within the page showing the publications of a DEI professor (see Fig. 6). Link “Details” in the left side navigation bar is used as a link to the DEI Member page. “Details” under each publication is instead used to display the detailed description of a particular publication. Interpreting the problem in the light of the Nielsen’s heuristics, we can therefore observe that:

1. The system-oriented language has been used instead of the user language. In fact, “Details” is the term that designers have being using during the application development to indicate the display of detailed contents of an information entity (such as DEI Member and Publication). This problem is therefore an example of a system that doesn’t speak the users' language, which can be solved by assigning links with names highlighting the actual semantics of contents displayed in the target page.

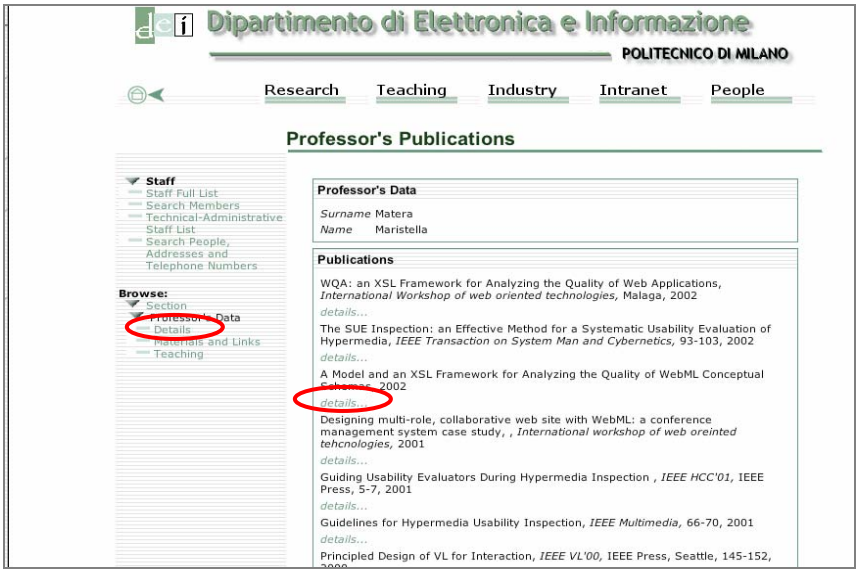

Fig. 6. Ambiguous semantics for the "details" link label

2. Adopting a same name for denoting two different concepts constraints users to "remember" the model of the implemented interaction, rather than "recalling" it. The interface doesn't make objects, actions, and options visible. The user has therefore to remember how to reach some contents across different application areas and different interaction sessions.

7 Concluding Remarks

Web applications are growing in number and complexity, becoming the de facto standard for distributed applications requiring human interaction. The ever-increasing spread of Web applications among not expert users, the abundance of their contents and the complexity of their hypertext interfaces, as well as their increased use for activities that range from everyday tasks to mission-critical actions determine a context in which usability is a relevant factor for the success of such applications.

The virtuous process of applying engineering techniques for enhancing the development of Web applications started a few years ago [5, 14, 57]: Web modeling techniques brought many benefits, but even though applying design principles is now widely accepted, the problem of poor

application design is still significant. Web engineering provides designers with a collection of tools and languages: tools speed up the development process, while formal specification languages enforce some sort of syntactic correctness and allow for (semi or complete) automatic code generation. Syntactic correctness prevents the designer from specifying an application containing flaws resulting in bugged code generation, but a quality application is much more than a bug-free piece of code.

Applications that incorporate usability engineering into their development process will be better able to comply with quality requirements. In particular, as reported in [38]:

1. Evaluation is the key for assuring quality: the effort put into evaluation directly determines the quality of the final applications.
2. In order to be effective the evaluation must relies over the establishment of suitable usability criteria to be verified.

This chapter has overviewed the most acknowledge methods currently adopted for the usability evaluation of Web applications, together with a set of criteria on which the evaluation of Web applications can be grounded. The chapter has also highlighted the most relevant advantages and drawbacks of the different methods, so as to guide practitioners into the choice of the most suitable method with respect to the evaluation goals.

Independently from the advantages offered by the adoption of a given method, it is worth noting that professionals and researchers suggest that a correct research plan for usability evaluation should usually cover the application of different techniques. The idea is that the characteristics of each method determine its effectiveness to discover a specific class of usability problems. Therefore different methods can be used in a complementary manner, so as to ensure a major completeness of the evaluation results.

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 developer 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 might allow developers to easily code and execute procedures for the verification of in-house guidelines, to make them easily enforced.

However, tools can help verifying structural properties, while they fail assessing all those properties that require judgements by human specialists, as well as providing answers about the nature of a discovered problem and suggestions about the needed design revisions. Automatic tools are

therefore very useful when their use complements the activity of human specialists.

8 References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: *ACM-SIGMOD'93*, ed by Buneman, P., Jajodia, S., International Conference on Management of Data, Washington, D.C., May 1993 (ACM, New York 1993) pp 207-216
2. Analog. http://www.analog.cx. (2005). Cited 18 Jan 2005
3. A-Prompt Project. http://aprompt.snow.utoronto.ca/ (2005). Cited 18 Jan 2005
4. AWSD-WebLog. http://awsd.com/scripts/weblog/index.shtml (2005). Cited 18 Jan 2005
5. Baresi, L., Garzotto, F., Paolini, P.: Extending UML for Modeling Web Applications. In: *HICSS'01*. 34th Annual Hawaii International Conference on System Sciences, Maui (USA), January 2001 (IEEE, 2001)
6. Berendt, B., Hotho, A., Stumme, G.: Towards Semantic Web Mining. In: *The Semantic Web - ISWC 2002*, ed by Horrocks, I., Hendler, J.A., First International Semantic Web Conference, Sardinia, Italy, June 2002. Lecture Notes in Computer Science, vol 2342 (Springer, Berlin 2002) pp 264-278
7. Berendt, B., Spiliopoulou, M.: Analysis of Navigation Behaviour in Web Sites Integrating Multiple Information Systems. VLDB J. **9**(1), 56-75 (2000)
8. Bias, R.G., Mayhew, D.J. (eds): *Cost-justifying usability* (Academic Press, Boston, MA 1994)
9. Blackmon, M. H., Polson, P.G., Kitajima, M., Lewis, C.: Cognitive walkthrough for the Web. In: *CHI'02*, International Conference on Human Factors in Computing Systems, Minneapolis, USA, April 2002 (ACM, 2002) pp 463-470
10. Bobby. http://bobby.watchfire.com/bobby/html/en/index.jsp (2005). Cited 18 Jan 2005
11. Brajnik, G.: Using automatic tools in accessibility and usability assurance. In: *ERCIM UI4ALL*, ed by Stephanidis, C., 8th International ERCIM Workshop on User Interface for All, Vienna, June 2004. Lecture Notes in Computer Science vol 3196 (Springer, Berlin 2004) pp 219-234
12. Brooks, P. Adding value to usability testing. In [51] (1994) pp 255-271
13. Ceri, S., Fraternali, et al.: Architectural Issues and Solutions in the Development of Data-Intensive Web Applications. In: *CIDR'03*, First Biennial Conference on Innovative Data Systems Research , Asilomar, USA, January 2003 (2003)
14. Ceri, S., Fraternali, P. Bongio, A., Brambilla, M., Comai, S., Matera, M.: *Designing Data-Intensive Web Applications*. (Morgan Kaufmann, San Francisco, CA 2003)

15. Ceri, S., Fraternali, P., Matera, M.: Conceptual Modeling of Data-Intensive Web Applications. IEEE Internet Computing **6**(4), 20-30 (2002)
16. Conallen, J.: *Building Web Applications with UML* (Addison Wesley, Boston 2002)
17. Cooley, R.: The use of Web Structures and Content to Identify Subjectively Interesting Web Usage Patterns. ACM TOIT **3**(2), 93-116 (2003)
18. Cooley, R., Mobasher, B., Srivastava, J.: Data Preparation for Mining World Wide Web Browsing Patterns. Knowledge and Information Systems **1**(1), 5-32 (1999)
19. Cooley, R., Tan, P., Srivastava, J.: Discovery of Interesting Usage Patterns from Web Data. In: *WebKDD'99*, ed by Masand, B.M, Spiliopoulou, M., International Workshop on Web Usage Analysis and User Profiling, San Diego, USA, August 1999. Lecture notes in Computer Science, vol 1836 (Springer, Berlin 2000) pp 163-182
20. Dix, A., Finlay, J., Abowd, G., Beale, R.: *Human-Computer Interaction*, 2nd edn (Prentice Hall, London 1998)
21. Doubleday, A., Ryan, M., Springett, M., and Sutcliffe, A.: A Comparison of Usability Techniques for Evaluating Design. In: *ACM DIS'97*, ed by van der Veer, G.C., Henderson, A., Coles, S., Symposium on Designing Interactive Systems: Processes, Practices, Methods and Techniques, Amsterdam, the Netherlands, August 1997 (ACM, New York, 1997) pp 101-110
22. Eirinaki, M., and Vazirgiannis, M.: Web Mining for Web Personalization. ACM TOIT, **3**(1), 1-27 (2003).
23. Fraternali, P., Lanzi, P.L., Matera, M. Maurino, A.: Model-Driven Web Usage Analysis for the Evaluation of Web Application Quality. Journal of Web Engineering **3**(2), 124-152 (2004)
24. Fraternali, P., Matera, M. Maurino, A.: WQA: an XSL Framework for Analyzing the Quality of Web Applications. In: *IWWOST'02*, Second International Workshop on Web-Oriented Software Technologies, Malaga, Spain, June 2002 (2002)
25. Garzotto, F., Matera, M.: A Systematic Method for Hypermedia Usability Inspection. New Review of Hypermedia and Multimedia, **6**(3), 39-65 (1997)
26. Hull, L.: Accessibility: It's Not Just for Disabilities Any More. ACM Interactions **41**(2), 36-41 (2004)
27. Hutchins E.L., Hollan J.D. and Norman, D.A.: Direct manipulation interfaces. Human-Computer Interaction, **1**, 311-338 (1985)
28. IBM. Ease of use guidelines. http://www-306.ibm.com/ibm/easy/eou_ext.nsf/publish/558 (2005). Cited 18 Jan 2005
29. ISO (International Standard Organization). ISO 9241: Ergonomics Requirements for Office Work with Visual Display Terminal (VDT)-Parts 1-17 (1997)
30. Ivory, M.Y., Hearst, M.A.: The state of the art in automating usability evaluation of user interfaces. ACM Comput. Surv. **33**(4), 470-516 (2001)

31. Ivory, M. Y., R. R. Sinha, M. A. Hearst: Empirically Validated Web Page Design Metrics. In: *ACM CHI'01*, International Conference on Human Factors in Computing Systems, Seattle, USA, April 2001 (ACM, New York 2001) pp 53-60
32. Jeffries, R., Desurvire, H.W.: Usability Testing vs. Heuristic Evaluation: Was There a Context? ACM SIGCHI Bulletin, **24**(4), 39-41 (1992)
33. Jeffries, R., Miller, J., Wharton, C., Uyeda, K.M. User Interface Evaluation in the Real Word: a Comparison of Four Techniques. In: *ACM CHI'91* - International Conference on Human Factors in Computing Systems, New Orleans, USA (ACM, New York 1991) pp 119-124
34. Kantner, L., Rosenbaum, S.: Usability Studies of WWW Sites: Heuristic Evaluation vs. Laboratory Testing. In: *ACM SIGDOC'97*, International Conference on Computer Documentation, Snowbird, USA (ACM, New York 1997) pp 153-160
35. Lewis, J.R.: IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instruction for Use. International Journal of Human-Computer Interaction, **7**(1), 57-78 (1995)
36. LIFT. http://www.usablenet.com (2005), Cited 18 Jan 2005
37. Lim, K.H., Benbasat, I., and Todd, P.A.: An Experimental Investigation of the Interactive Effects of Interface Style, Instructions, and Task Familiarity on User Performance. ACM Trans. on Computer-Human Interaction, **3**(1), 1-37 (1996)
38. Lowe, D.: Emerging knowledge in Web Development. In *Managing Software Engineering Knowledge*, ed by Aurum, A, Jeffery, R, Wohlin, C, Handzic, M. (Springer, Berlin 2003) pp 157-175
39. Lynch, P., Horton, S.: *Web Style Guide: Basic Design Principles for Creating Web Sites*, 2nd edn (Yale University, 2001)
40. Madsen, K.H.: Special Issue on "The Diversity of Usability Practices". Comm. ACM, **42** (5) (1999)
41. Marchionini, G., Shneiderman, B.: Finding facts vs. browsing knowledge in hypertext systems. IEEE Computer, **21** (1), 70-80 (1988)
42. Meo, R., Lanzi, P.L., Matera, M., Esposito, R.: Integrating Web Conceptual Modeling and Web Usage Mining. In: *WebKDD'04*, International ACM Workshop on Web Mining and Web Usage Analysis, Seattle, USA, August 2004 (2004)
43. Molich, R., Nielsen, J.: Improving a Human-Computer Dialogue. Comm. ACM, **33**(3), 338-348 (1990).
44. Nielsen, J.: The Usability Engineering Lifecycle. IEEE Computer, **25**(3), 12-22 (1992)
45. Nielsen, J.: *Usability Engineering* (Academic Press, Cambridge, MA 1993)
46. Nielsen, J.: Special Issue on "Usability Laboratories". Behavior and Information Technology, **13**(1) (1994)
47. Nielsen, J. Guerrilla HCI: Using Discount Usability Engineering to Penetrate Intimidation Barrier. In: *Cost-Justifying Usability*, ed by Bias, R.G., Mayhew, D.J. (Academic Press, Cambridge, MA 1994).

48. Nielsen, J.: *Multimedia and Hypertext. Internet and Beyond* (Academic Press, London 1995)
49. Nielsen, J.: *Web Usability* (New Riders, New York 2000)
50. Nielsen, J., Landauer, T.K.: A Mathematical Model of the Finding of Usability Problems. In: *ACM INTERCHI'93*, International Conference on Human Factors in Computing Systems, Amsterdam, NL, April 1993 (ACM, New York 1993) pp 296-213
51. Nielsen, J., Mack, R.L.: *Usability Inspection Methods*. (Wiley, New York 1994)
52. Nielsen, J., Molich, R.: Heuristic evaluation of user interfaces. *In: ACM CHI'90*, International Conference on Human Factors in Computing Systems, Seattle, USA, April 1990 (ACM, New York1990) pp 249-256.
53. Norman, D.A.: Cognitive artifacts. In: *Designing Interaction: Psychology at the Human–Computer Interface*, ed by Carroll, J. M. (Cambridge University, New York 1991) pp. 17–38
54. Polson, P., Lewis, C., Rieman, J., Wharton, C.: Cognitive Walkthrough: A Method for Theory-based Evaluation of User Interfaces. International Journal of Man-Machine Studies, **36**, 741-773 (1992)
55. Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., Carey, T.: *Human-Computer Interaction* (Addison Wesley, New York 1994)
56. Punin, J.R., Krishnamoorthy, M.S., Zaki, M.J.: LOGML: Log Markup Language for Web Usage Mining. In: *WebKDD'01- Mining Web Log Data Across All Customers Touch Points*, ed by Kohavi, R., Masand, B.M, Spiliopoulou, M. Srivastava, J., Third International Workshop on Web Mining and Web Usage Analysis, San Francisco, USA, August 2001. Lecture Notes in Computer Science vol 2356 (Springer, Berlin 2002) pp 88-112
57. Schwabe, D., Rossi, G.: An Object Oriented Approach to Web-Based Applications Design. TAPOS **4**(4), 207-225 (1998)
58. Shneiderman, B.: *Designing the User Interface. Strategies for Effective Human-Computer Interaction* (Addison Wesley, New York 1992)
59. Shneiderman, B.: Universal Usability. Communication of ACM, **43**(5), 84-91 (2000)
60. Shneiderman, B., Byrd, D., Croft, W.B.: Sorting out searching. Communication of ACM, **41**(4), 95-98 (1998)
61. Srivastava, J., Cooley, R., Deshpande, M., Tan, P.N.: Web usage mining: Discovery and applications of usage patterns from Web data. SIGKDD Explorations, **1**(2), 12.23 (2000)
62. Stroulia, E., Niu, N., El-Ramly, M.: Understanding Web usage for dynamic Web-site adaptation: A case study. In: *WSE'02*, Fourth International Workshop on Web Site Evolution, Montreal, Canada, October 2002 (IEEE, 2002) pp 53-64
63. Theofanos, M.F., Redish, J.: Bridging the gap between accessibility and usability. ACM Interactions, **10** (6), 36-51 (2003)
64. Virzi, R.A.: Refining the test phase of usability evaluation: How many subjects is enough? Human Factors, **34(**4), 457-468 (1992)

65. WebCriteria SiteProfile. http://www.coremetrics.com (2005). Cited 18 Jan 2005
66. WebRatio Site Development Studio. http://www.webratio.com (2005) Cited 18 Jan 2005
67. Wharton, C., Rieman, J., Lewis, C., Polson, P.: The Cognitive Walkthrough Method: A Practitioner's Guide. In [51] (1994), pp 105-140.
68. Whiteside, J., Bennet, J., Holtzblatt, K.: Usability Engineering: Our Experience and Evolution. In: *Handbook of Human-Computer Interaction*, ed by Helander, M. (Elsevier, 1988) pp 791-817
69. Wilson, T.D.: Human Information Behavior. Informing Science, **3**(2), 49-55 (2000)
70. Wurman, R. S.: *Information Architects* (Watson-Guptill, New York 1997)
71. W3C Consortium - Extended log file format. http://www.w3.org/TR/WD-logfile.html (2003). Cited 18 Jan 2005
72. W3C Consortium - WAI-Web Content Accessibility Guidelines 2.0. W3C-WAI Working Draft. http://www.w3.org/TR/WCAG20/ (2004). Cited 18 Jan 2005