Towards Automated Testing of Web Usability Guidelines

Dominique Scapin and Corinne Leulier
Dominique.Scapin@inria.fr, Corinne.Leulier@inria.fr
Institut National de Recherche en Informatique et en Automatique
Unité de recherche Rocquencourt, Domaine de Voluceau – B.P. 105, F-78153 Le Chesnay Cedex, France
Web: http://www.inria.fr/Dominique.Scapin

Jean Vanderdonckt and Céline Mariage
vanderdonckt@qant.ucl.ac.be, mariage@qant.ucl.ac.be
Université catholique de Louvain (UCL), Place des Doyens, 1 – B-1348 Louvain-la-Neuve, Belgium
Web: http://www.qant.ucl.ac.be/membres/jv

Christian Bastien
Christian.Bastien@ergo-info.univ-paris5.fr
Laboratoire d'Ergonomie Informatique, Université René Descartes, 45, rue des Saints-Pères – F-75270 Paris Cedex 06, France
Web: http://www.univ-paris5.fr/LEI

Christelle Farenc, Philippe Palanque, and Rémi Bastide
Farenc@univ-tlse1.fr, Philippe.Palanque@univ-tlse1.fr, Remi.Bastide@univ-tlse1.fr
Université Toulouse I, UFR d'Informatique, Laboratoire d'Interaction Homme-Système (L.I.H.S), Place Anatole France – F-31042 Toulouse, France
Web: http://lis.univ-tlse1.fr/~farenc/ , http://lis.univ-tlse1.fr/palanque/, http://lis.univ-tlse1.fr/bastide/

1. Introduction

Usability of web sites, whatever their type is (e.g., internet, intranet, extranet, supranet) is today widely recognized as a major dimension required for user acceptance and use of web sites. This dimension even becomes more critical for electronic commerce web sites: a customer unsatisfied by poor usability is likely to become a competitor's customer. Moreover, the user population is expanding in age (ranging from young users to elderly people), in expectations (ranging from private use for leisure to professional use), in information needs (ranging from simple information to compound multimedia resources), in task types (ranging from basic text searches to complex problem-solving methods), and in user abilities (ranging from the able-bodied person to any person with special needs, such as for motor-, auditory- or visually-impaired persons). Therefore, web sites should be usable enough to accommodate all these variations over time.

Numerous evaluation methods help ensure that web sites are usable (Nielsen,94; Nielsen & Mack,94): inquiry methods, inspection methods, user testing, to name a few categories of them. These methods are so numerous and hard to differentiate that identifying which ones are suitable for a particular case study is challenging. This identification is often practically constrained by several needs, as:

- The need for a cost-effective method: it is often heard that project resources in time, budget, and personal are so limited that they prohibit any use of evaluation method (Lynch, Palmiter & Tilt,98).

- The need for remote evaluation (Hartson et al.,96): to reach a wider audience and to cover multiple evaluation circumstances, evaluators may be separated in time and/or space from the users.
- The need for challenging evaluations (Cooper,99b), such as very large web sites and dynamic web sites: very large web sites are characterized by being any large-scale, information abundant, interactively rich web site installed in a distributed environment (e.g., in different physical locations) with heterogeneous software and hardware (e.g., on different servers). They are often designed by a wide variety of person having little or no experience on usability engineering. Yet they have access to powerful web development tool, but almost none of them incorporate any form of user guidance regarding usability. When a web site basically consists of pages generated dynamically on the fly according to different parameters, evaluating all possible combination of pages becomes impossible. Rather, a sample enhanced by user-related data should be considered. These two cases are representative examples of situations where performing an evaluation is challenging because of the size, the complexity, or the dynamics.
- The need for real-time evaluation: when a web site is rapidly changing over time, such as for dynamic web site, the organization often wants to keep the evaluation accurate and up to date with respect to the current state of the web site. In this case, performing large-scale or expensive evaluations is prohibited and incompatible with the rapidly changing nature.
- The need for raising the usability awareness of designers and evaluators (Scholtz, Laskowski & Downey, 98): teaching usability to designers who are unaware of it as they use it or improving the experience of those who already are aware of it can be a very valuable process; their usability awareness and their skills to conduct it are likely to grow up as their experience is accumulated.
- The need for reusable evaluation results: an evaluation method should produce tangible results that should be interpreted, processed in a straightforward manner so as to reuse them when the web site is updated or improved. If an evaluation method does produce results that designers cannot practically consider or interpret, the method is doomed to failure. For this purpose, concrete guidelines should be preferred to general guidelines.

All these needs generally plaid for any evaluation method that is easy to use, cheap to manipulate, and quick to conduct. Therefore, rapid lightweight methods supported by software tools automating it partially or completely are often considered relevant. The automation is highly desirable when designers or evaluators need to be released from repetitive, tedious tasks to focus on task-based aspects.

For these reasons, our work is focusing on the use of web usability guidelines in heuristic evaluation. On the one hand, guidelines have become a popular form for conveying some usability wisdom that can be expressed, communicated, and propagated effectively and efficiently; on the other hand, a profusion of sources containing guidelines is available, letting thus believe that these resources can provide a valuable starting point for conducting heuristic evaluation. We define a *usability guideline* as any statement ensuring some adequacy of a particular user interface of a computer-based system (here, a web site) with respect to a particular context of use where a given user population has to fulfill interactive tasks with a given system in certain situations.

Respecting these guidelines have already been proved to have a positive impact on usability (Comber,95; Grose, Forsythe & Ratner, 98). For instance, Borges *et al.* showed that the average time to carry out 5 tasks on a web site respecting 17 guidelines has been reduced by at least by 16% (Borges, Morales & Rodríguez,96). But this does not necessarily imply that a web site not addressing guidelines is unusable. Nor it is proved that a web site addressing all guidelines is the most usable site. In other words, guidelines are necessary, but insufficient, and should not be considered in isolation. Often

guidelines need to be supplemented by a suitable method and a clear process that leads them to unambiguous interpretation.

Although guidelines can be used in conjunction with guidelines checklists, standard inspection, consistency inspection, we chose a variant of heuristic evaluation where the usual set of 10 heuristics (Nielsen,94) is replaced by a set of specific guidelines. We did this because heuristics, albeit cost-effective, have been considered too general and not expressive enough to be evaluated automatically. Conversely, lists of guidelines are potentially long and specific, taking a lot of time to evaluate them since the whole user interface and its components should be evaluated against them. But it is hoped to automate as far as possible this process. Our main goal in this work is consequently:

- To define a framework that facilitates the structuring and the use of guidelines for design and evaluation.
- To develop a tool that exploits this framework to provide automated testing of web sites against guidelines structured thanks to the framework.

The rest of this paper is structured as follows: section 2 reports on the usability knowledge contained in web usability guidelines, where they come from, what are some of their shortcomings and review some software tools for working with these guidelines. A particular focus on automation will guide this review. Section 3 summarizes five development milestones needed to obtain such a tool. As one of our goals is to develop a tool for automated testing of guidelines, we will discuss how these milestones can be browsed to reach such a tool, and what our approach is in achieving these milestones. Section 4 will exemplify the whole process of selected guidelines, sorted by order of complexity towards their automated testing. As several other evaluation method provide some automated evaluation, at least partially, we briefly review them in section 5 with respect to our current approach for automated testing. Section 6 will outline several dimensions towards which this approach could evolve and will detail some of them that have been selected for future work. We conclude by emphasizing the need for a global theory for systematic and rigorous evaluation method based on guidelines.

2. Web Usability Guidelines

In this section, possible sources for collecting web usability guidelines are firstly indicated. Secondly, some shortcomings of guideline usage are discussed as they mostly affect the results produced by heuristic evaluation. Their level of expressiveness is one major reason of concern. Thirdly, two categories of software tools for working with guidelines will be introduced and exemplified. The discussion of this entire section is valid for general guidelines. However, as this paper is dedicated to web usability guidelines, an emphasis will be put on relevant aspects.

2.1 Sources for Web Usability Guidelines

Many guidelines have been published in document sources throughout the literature. These sources fall into five categories (Vanderdonckt,99):

1. *Design rules*: they comprise a set of functional and/or operational specifications that specify the design of a particular user interface. These specifications are presented in a form that requires no further interpretation, either from designers or from developers. Their straightforward format allows an immediate exploitation (Fig. 1).

2. *Compilations of guidelines*: they comprise several prescriptions written for a wide range of user interfaces. Each prescription is presented as a statement, sometimes along with examples, with or without clarifying explanations and comments. Each prescription generally results from a human consensus between guideline users. This consensus is less relevant once a prescription is experimentally tested and verified. They can range from a small set of guidelines dedicated to a particular usability feature (e.g., interaction technique) to an extensive collection of guidelines covering a family of tasks and domains. For instance, guidelines for multimedia, interactive kiosks, accessibility (Vanderheiden et al., 97), for web site in general (Detweiler & Omanson,96). Some of these are validated by experimental results provided by user testing, laboratory experiences or others.
3. *Style guides*: they comprise a set of guidelines and/or functional or non-functional specifications aiming at consistency for a collection of distinct user interfaces. This collection can be based on an operating system, on a software editor, by a particular physical environment such as SUN (Levine,96), by a domain of human activity or by a corporate. In particular, Ratner, Grose & Forsythe (1996) identified that only 20% of HTML relevant recommendations from established style guides were found in HTML style guides.
4. *Standards*: they comprise a set of functional and/or operational specifications intended to standardize design. Standards are promulgated by national or international organizations for standardization. They can be military, governmental, civil or industrial. There is no web dedicated standard we are aware of, but some important general standards are good candidates to tailor their guidelines to suit the exact issues faced by web sites, e.g., ISO 9241 (ISO,99a,b), HFES/ANSI 200 (Hfes,97).
5. *Ergonomic algorithms*: these aim to systematize one design aspect by building it in the same way, according to a same base of design rules. They appear as a software component that implements an algorithm rather than a paper procedure. Such algorithms are primarily intended to design a series of web pages that automatically respect some guideline (usually, design rules or style guides) by construction. In this way, they immediately enforce the address of guidelines without the need to take care of them. Of course, cascading style sheets or page templates can be resourceful, provided that their quality intrinsically mirrors the application of guidelines, which is not necessarily the case. For example, DON (Kim & Foley,93) is capable of automatically laying out widgets to minimize the screen space, while preserving label and fields alignments to fasten the reading path. We observed that we have not yet seen a true software providing designers with some assistance in designing usable pages, but they may come.

Contrarily to the domain of usability for graphical user interfaces, web usability guidelines in these categories are often produced by consensus, common sense or observation of best practices. Guidelines validated by experimental results are rare and hard to find out. Moreover, web usability guidelines tend to address web specific usability issues, but not necessarily usability issues for graphical user interfaces adapted or specialized for the web. These observations also formed one of our motivations: usability guidelines that are valid for graphical user interfaces must also be considered as potential usability guidelines for the web, unless some contradiction or invalidation is appearing. Only categories of rigorous sources providing experimentally validated guidelines are considered the most reliable. Such sources have been selected, while others have been discarded for this reason.

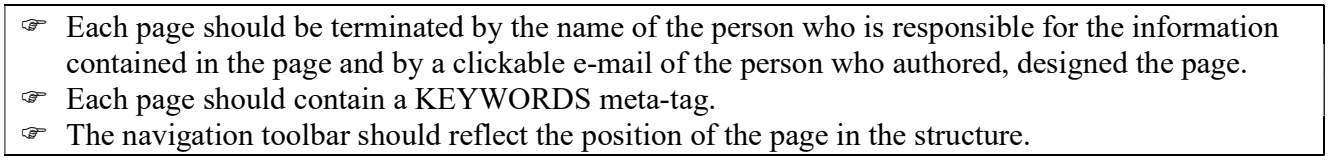

Fig. 1. Examples of design rules.

2.2 Some shortcomings for guidelines

Despite web usability guidelines have been proved useful as indicated, they still suffer from a series of shortcomings that impede their use and significantly reduce their scope. An extensive discussion can be found in (Vanderdonckt,99). We only focus here on some of them that are relevant for web sites.

Firstly, the level of guideline expressiveness and the confidence in applying them highly depends on the source where the guidelines come from. Fig. 2 depicts that guidelines found in the five types of sources can range from a very abstract, yet general, form to a very concrete, yet specific, form. The more abstract guidelines are, the more general they become and the larger scope of domain they can be considered for. Conversely, as guidelines become very concrete, their application is often restricted to given contexts of use. Various names are used to refer to general guidelines: rules of thumb, usability factors, dimensions, and principles.

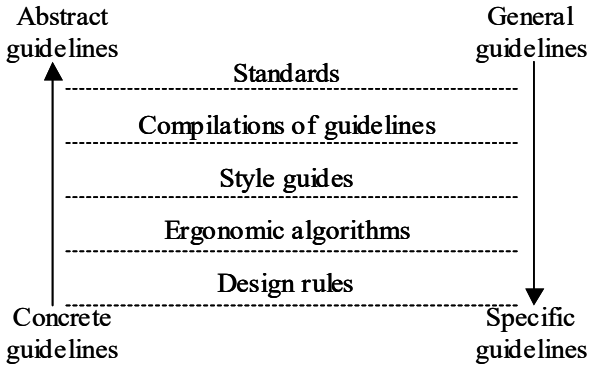

Fig. 2. Location of sources containing guidelines.

Secondly, as a consequence of the previous observation, abstract guidelines cannot be applied per se, thus requiring some interpretation to become concrete for the intended context of use. On one hand, this abstraction can be benefit to put aside experimental conditions under which the guideline has been tested and validated; on the other hand, these conditions, which are required to ensure a correct interpretation, may invalidate any such interpretation as they vanished. Concrete guidelines no longer require such interpretation, but are so specific that they prevent designers to apply them successfully in other situations without any risk of invalidity. Therefore, too high-level guidelines are difficult to formulate and quantify when they need to be applied or evaluated. Test automation without formalization of guidelines is doomed to failure (Farenc, Liberati & Barthet,96).

Thirdly, the jargon or specific vocabularies used in the sources coming from various disciplines (e.g., cognitive modeling, psychology, human factors, ethnography) may also prevent designers to appropriately understand them and applying them correctly. Once again, extensive relevant experience may be needed to avoid incorrect abstraction or concretization of guidelines that will invalidate their results.

Fourthly, guidelines can be classified according to their linguistic level, i.e. goal, pragmatic, semantic, syntactic, lexical, alphabetical, and physical. Guidelines located at the lower levels of this layered model are easier to interpret and to apply than those located at the higher levels.

Fifthly, the workload involved by evaluating a single guideline is depending on several properties of the guidelines, namely, but not limited to: its linguistic level, the quality of its statement, and their

scope. A guideline can be relevant for one individual page at a time (Fig. 3a), for a series of related web pages (Fig. 3b), and for the whole site (Fig. 3c).

- ☞ Since most viewers will use 13” colors monitors, make sure your web page fit comfortably within the parameters of these smaller displays (Detweiler & Omanson,96).
- ☞ Give users control to specify the extent of their searches, e.g., locally within a section or subsection (Detweiler & Omanson,96).
- ☞ Help orient users by establishing useful expectations about the site’s organization and/or functionality (Detweiler & Omanson,96).

Fig. 3. Scope of different guidelines **(a,b,c)**.

2.3 Existing Tools for Working with Guidelines

Two categories of tools exist to provide assistance to designers:

1. *Passive tools*: this category only provides designers with powerful access to guidelines, but these guidelines are in no way executable in any sense. Facilities can be offered to gather guidelines, to select them, to build a report for evaluation purposes, or for documentation, illustration or teaching.
2. *Active tools*: this category provides designers with tools capable of some form of processing guidelines, either at design time or at evaluation time. Representative tools belonging to this category are reported in (Vanderdonckt,99). In particular for the Web, there are validation checkers concentrating of the HTML Code (Clark & Dardailler,99), such as WebLint (Bower,96), HTML Validator (Yahoo,2000), but these verifications, although useful, are not related to usability. To our knowledge, only three tools are devoted to some form of automated testing of web usability guidelines:

    - Bobby (Cooper,99a;b), from CAST, automatically checks a web page or a series of web pages against accessibility guidelines promoted by the Web Accessibility Initiative (WAI) (WAI,98). Clicking on any Bobby hat appearing on the resulting page leads to a reference where the problem is detected and some comment on the guideline (fig. 4). WAI guidelines have been considered for implementation and evaluation through Bobby with three different levels of support: fully automated, partially automated, and manual, 49% of these shown non-manual.

Fig. 4. Location of accessibility errors identified by Bobby on a Web page.

- WebSAT (Scholtz, Laskowski, and Downey,98), from the WebMetrics Suite (Scholtz,98) automatically test a single or a series of web pages against a predefined set of guidelines that are not necessarily focusing on accessibility. As Bobby, a static analysis of the HTML code is performed and submitted to usability testing by a formal approach. Results are then summarized into six categories (fig. 5): accessibility (e.g., "All images not used as links should contain ALT tags"), form use (e.g., "The form should include a functionality for returning the completed form"), performance, maintainability, navigation, and readability (e.g., "Try to limit the density of the web page"). Similarly, explanation for each potential detected problem can be provided, as well as a check list of them.

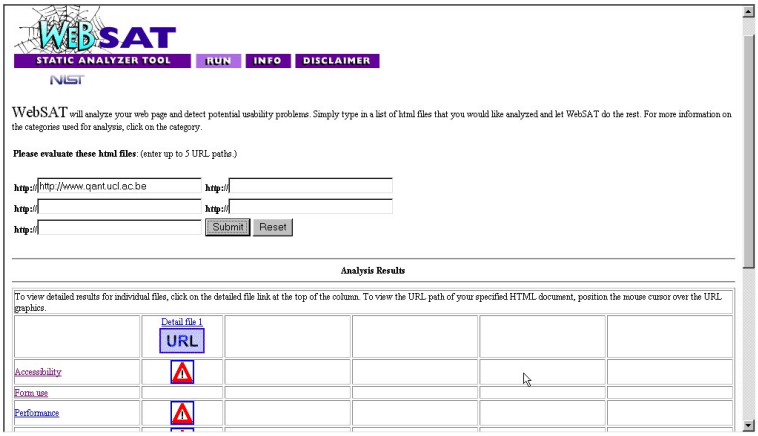

Fig. 5. Location of usability errors identified by WebSAT on a Web page.

- Design Advisor (Faraday,99) automatically critiques a web page as the designer is modifying its elements. The critique is based on a visual hierarchy of perceptual guidelines assuming that page elements are searched according to a priority order: motion, size, images, color, text style, and position. From these guidelines, the tool automatically superimposes a scanning path on the web page, thus highlighting usability problems (fig. 6). Although this tool is primarily based on these specific guidelines, it is interesting to note here that they have been implemented as Prolog clauses and the attributes of web elements as facts. Having such an inference engine would be very practical for us, as adding, deleting, or modifying any guideline in the knowledge base would have no effect on their execution. But so far we are unsure that all guidelines can be restricted to Prolog clauses. A deeper understanding of these possible restrictions, where any, is required. It is not certain that all guidelines can be implemented this way.

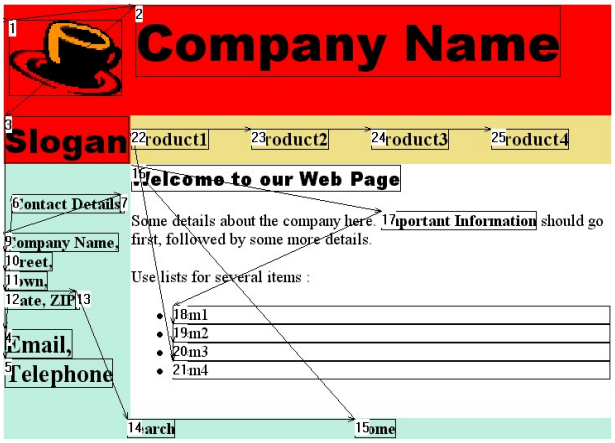

Fig. 6. Scanning path superimposed on a web page by Design Advisor.

Our main goal is to produce a tool for automated testing of usability guidelines as far as they can be implemented. For this purpose, a specific tool for working with guidelines is needed. The next section explains how this goal can be reached.

3. Development milestones

To develop any tool for working with guidelines, five development milestones have been identified for reference and comparison purposes (Vanderdonckt, 99), but also for structuring the process to reach that goal. These steps are: guidelines collection, guidelines organization, incorporation of guidelines into approach, guidelines operationalization, and guidelines use.

3.1 Guidelines collection

The goal of this first step is to gather a useful subset of guidelines suitable for designing web pages. In this study, three sources containing guidelines have been selected (Mayhew,92; ISO,99b; Vanderdonckt,94) as they each consist in a somewhat extensive compilation of guidelines (respectively, 293, 488, more than 3,000!). In the ISO general standard, only Part 12 “Presentation of Information”, Part 13 “User guidance”, Part 14 “Menu dialogues”, Part 15 “Command dialogues”, Part 16 “Direct manipulation dialogues”, and Part 17 “Form-filling dialogues” have been selected. These different sources are not especially dedicated to web usability. Rather, they focus more on graphical user interface in general, thus requiring some modification, adaptation, extension, and so forth (Ratner, Grose & Forsythe,96). However, a previous compilation of web design guidelines was merged with these sources. Due to this high amount of guidelines considered and due to the high redundancy from one source to another, only unique guidelines have been kept. This process abundantly relied on generalization and specialization relationships, as a specific guideline found in one source often appeared as a specialization of a more general guideline found in another source. 466 guidelines resulted from this preliminary work.

3.2 Guidelines organization

This initial set of guidelines is copious and ranges over many levels of rigor and credibility. Both can be ameliorated by a good organizational structure, which is the main goal of guidelines organization. The guidelines organization basically involves two steps:

1. **Classifying each guideline by ergonomic criteria**. An *ergonomic criterion* is hereby defined as a well-recognized usability dimension in human-computer interaction whose reliability effectiveness and impact on usability have been experimentally assessed (Bastien & Scapin, 95; Scapin & Bastien, 97). These criteria have been successfully used for classifying web usability guidelines, too (Bastien, Scapin & Leulier, 99). Table 1 reproduces these ergonomic criteria. Leaf nodes in the decomposition are called elementary criteria as they are not further decomposed into sub-criteria. Other criteria that can be decomposed into other sub-criteria are called composite criteria. Each guideline has therefore been classified to a unique ergonomic criterion based on its definition. For this purpose, guidelines have been submitted to operations like unfolding and coupling into two, three or even four parts. Indeed, the statement of some guidelines remained too general or was condensing several guidelines at the same time. Therefore, to individualize each guideline, these operations have been performed and assessed by a panel of experts. Some guidelines have been affected to some criteria by consensus between the judging parties. The guidelines pool has been augmented by 74 individual guidelines, thus totalizing 540 resulting guidelines (Scapin et. al., 99). Fig. 7 graphically depicts the distribution of resulting guidelines by elementary criteria. It is worth to note here that the compatibility criterion received the most impressive attention by having the largest amount of resulting guidelines. The "Prompting" and the "Compatibility" criteria contain 69 individual guidelines, respectively 134, whereas "Explicit actions" and "User experience" only have 3, respectively 7 guidelines. This does not necessarily mean that these criteria are unimportant for designing and/or evaluating web sites, but that they are understated in the guidelines set. Classifying each guideline by ergonomic criteria goes beyond the simple indexation easiness: it can provide designers with a first idea on when and where the related guideline can be applied as well as some first idea of their absolute level of importance. For instance, a guideline related to task compatibility may be considered more important than a guideline related to grouping items by format.

1. Guidance
1.1. Prompting (PROM)
1.2. Grouping / Distinction between items
1.2.1. Grouping / Distinction by location (GDLO)
1.2.2. Grouping / Distinction by format (GDFO)
1.3. Immediate feedback (FEED)
1.4. Legibility (LEGY)
2. Work load
2.1. Brevity
2.1.1. Concision (CONC)
2.1.2. Minimal actions (MIAC)
2.2. Information density (INDE)
3. Explicit control
3.1. Explicit actions (EXUA)
3.2. User control (USCO)
4. Adaptability
4.1. Flexibility (FLEX)
4.2. User experience (USEX)
5. Error management
5.1. Protection to errors (ERPR)
5.2. Quality of error messages (QUEM)
5.3. Error correction (ERCO)
6. Homogeneousness / Consistency (CONS)
7. Significance of codes (SICO)
8. Compatibility (COMP)

Table 1. Hierarchy of ergonomic criteria and sub-criteria.

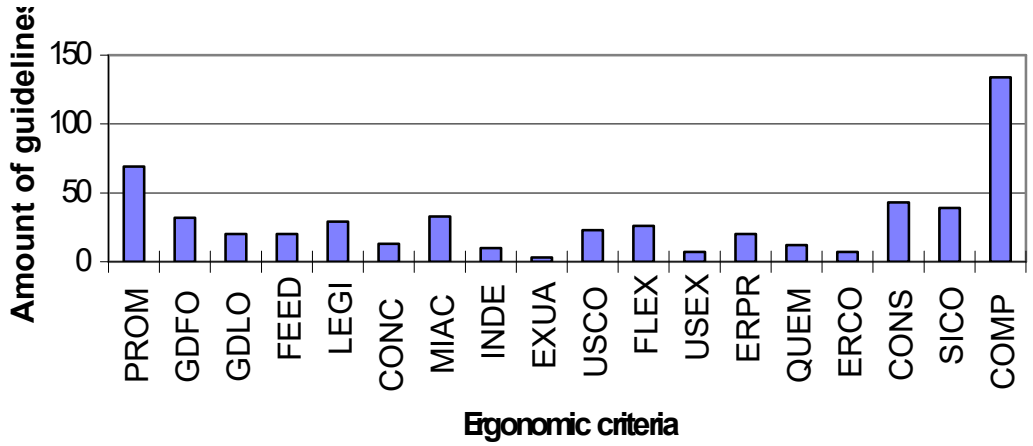

Fig. 7. Distribution of guidelines by elementary ergonomic criteria.

2. **Further classifying each guideline:** as the resulting set of guidelines is still huge, guidelines are further classified by alternate index keys. This classification allows multiple and flexible access paths to each guideline, rather than merely by ergonomic criteria (Bastien, Scapin & Leulier,99). Such accesses permit automatic identification of guidelines for a particular page element (e.g., for an image map), for a particular design aspect (e.g., information layout), or any combination (e.g., how to lay out information in an image map). Such identifications are required for specific evaluation targets and/or to make guidelines applying directly linked to design options, thus making the evaluation process more constructive towards the ends of designers. These identifications also improve the access efficiency to guidelines: when a given page element is modified across a set of pages, corresponding guidelines can be evaluated more rapidly as they are specific to the context. Each guideline is assigned to one or many index keys relevant for web sites (Table 2).

Actions (Seq. of)	Image maps/hot areas	Plug-in
Alignment	Image maps/linked items	Printing
Animations/Video	Images/description	Radio buttons
Buttons/Labels	Information Layout	Reading task
Buttons/Position	Information/Content	Screen Layout
Check Boxes	Information/Organisation	Scrolling
Colours	Items organisation	Search Facilities
Colours/Black & White displays	Items/Presentation	Site maps
Downloading considerations	Language	Sounds
Errors	Lines	Tables
Forms	Links	Terms
Forms/Buttons	Links/Organisation	Terms/Graphics
Frames	Links/visual codes	Text
Graphics	List	Text & background
Graphics/Description	List/items	Text/Buttons
Graphics/Downloading	List/Structure	Text anchors
Graphics/Layout	Menus	Text colours
Graphics/Size	Menus/Scrolling lists	Text fields
Graphics/Text	Menu/selected items	Text-only Browsers
Grids	Metaphors	Text-only Browsers/Lines
Headings	Navigation structure	Text/Headers & footers
Help Facilities	Navigational aids	Title
Helper Applications	Navigational aids/Links	URL
Icons	Page Length	Vocabulary
Icons/Image maps	Page Length/Links	Windows
Icons/Labels	Page size	
Image maps	Page structure	

Table 2. Taxonomy of additional index keys.

3. Each guideline is then assigned to one or many evaluation methods (e.g., user questionnaire, user testing, user observation, cognitive walkthrough, card sorting) that can be effectively used to assess the guideline. In particular, each guideline is examined to identify the extent to which its evaluation can be software supported, preferably with automation. Five levels are distinguished: *full automatic* (when the guideline can be evaluated by the system), *semi-automatic* (when the guideline can be evaluated partly by the system), *questions-based* (when the system is asking questions to the evaluator to achieve the evaluation), *delegated* (when external methods or tools can provide evaluation support), or manual (when the guideline can be evaluated by the user only). Generalization/specialization relations are then exploited to establish more precise links between related guidelines. Finally, a score is attached to each guideline to express the impact the guideline can raise being respected or violated (fig. 8).

☞ Guideline: In order to support different levels of user skills, direct manipulation dialogues should be designed to minimize the need for users to alternate between different input devices.
Example: To fill in a form a user points and selects every field with the mouse and then enters text with the keyboard. As the experience grows the user moves the cursor from field to field with the tab key before entering text. Thereby the need for the user to alternate between input devices is minimized and efficiency is increased.
Source: Ergonomic requirements for office work with visual terminals (VDTs) - Part 16: Direct-manipulation dialogues, ISO/DIS 9241-16.
Guideline Number: (5.4.3) - Criteria: Users' experience - Index Key: Dialogue
Evaluation method: user observation (manual) – Score: rather important.

Fig. 8. Example of resulting guideline.

3.3 Incorporation of guidelines into approach

Early consideration of web usability guidelines in the development life cycle of the web site requires an iterative and eager design process. It is thus important to associate sections of guidelines with the different phases of a development life cycle. In this way, it is expected that sets of guidelines suitable for each phase can be more easily identified and accessed. The goal of this step is to locate points within these phases where organized guidelines should be considered, to specify which should be considered by identifying local guidelines (for a phase), global guidelines (for all phases) or pervasive ones (for several continuous phases). For this purpose, the general development life cycle illustrated in fig. 9 was used. Each guideline is consequently referred to any phase and sub-phases where suitable.

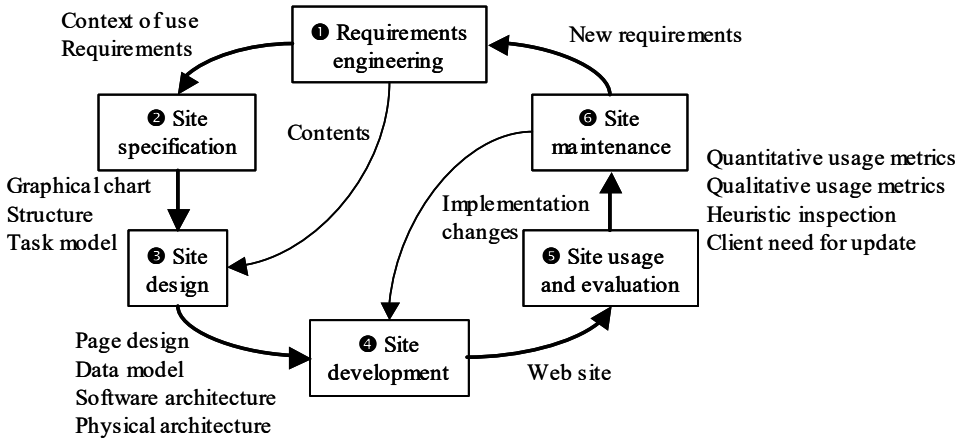

Fig. 9. General development life cycle for a web site.

3.4 Guidelines operationalization

Guidelines incorporated into a development life cycle are typically used manually. To embody them in a software tool, a further stage of guidelines operationalization is thus aimed at re-expressing in a more formal way (e.g., Harrison & Thimbleby, 85) guidelines that can be directly evaluated through a software tool. For this purpose, each guideline is formalized in a more systematic way by referring to the terms of a framework partially reproduced in Appendix 1. The aim of this framework is to enable expressing guideline with common scheme and vocabulary that should speed up human interpretation of any guideline and should clarify a procedure that can be effectively implemented for automated testing, where possible. Another advantage of this framework relies in its ability to identify information required to apply the guideline at design time end/or to assess it at evaluation time. The ultimate goal of this being that the complete procedure is described in the terms of the framework. The next section exemplifies how some guidelines can be expressed and automated thanks to this framework.

3.5 Guidelines use

We have no good knowledge about the usability of software tools for working with guidelines. And yet, it seems fundamental for HCI researchers that the UI of software tools for working with guidelines is usable. An empirical study of the usability of such tools would be of great benefit.

4. Some examples of automated testing of usability guidelines

4.1 Design rules

Let us imagine two design rules for the inclusion of a particular graphic:

- ☞ Each personal home page should contain the company logo at the top left corner.
- ☞ Each personal home page should contain the photo of the person after the logo.

According to the framework, the first guideline could be formalized as follows:

1.1.1.5 Specific tasks
  1.1.1.1.5.4 Search for personal home page /* definition of a particular task */
1.1.2 Information needed
  1.1.2.1. Nature: company logo /* definition of a domain object */
2.1.1.1 Specification C.E.-Compatibility-tasks /*First ergonomic criteria to be respected */
  2.1.1.1.1 Structure task: task should begin with displaying logo
2.1.2.5. Specification C.E.-Guidance /*Second ergonomic criteria to be respected*/
  2.1.2.5.1.3.2.4 Picture: picture=company logo
  2.1.2.5.2 Specification C.E.-Grouping/distinction
    2.1.2.5.2.1 Spec. C.E.-By location: location=left corner

The second can be formalized similarly. Capture-and-Replay testing tools (CR tools) represent some first attempt to automate compliance with design rules by performing four steps (Linz & Daigl,99):

1. *Capturing*: the tool captures static properties of objects in page (e.g., color, size) as well as dynamic properties (e.g., activation, deactivation).
2. *Programming*: the properties are stored in a test script to be programmed with traditional algorithmic structures.
3. *Inserting checkpoints*: checkpoints regarding properties are inserted into the script.
4. *Replaying*: the test script is run on any page to be evaluated and any discrepancy with respect to reference properties and checkpoints is detected.

Several design rules can be tested this way, such as “Ok and Cancel are always located at the bottom or on the right, but never on top or on the left”, “Submit and Reset push buttons in forms, if any, should be presented in this order”. Fig. 10 reproduces a screen capture where the two above design rules have been captured and subsequently replayed with Watchfire’s MacroBot (http://www.watchfire.com/-products/macrobot). It basically consists of two test cases testing the presence of the graphics at their right location, plus the general composition. In this case, the two guidelines are not respected.

CR tools are dependent on the very specific parameters of a guideline. In the recording process, it recorded the index position of an image the user pointed to in the images array rather than more uniquely identifying information such as its source. Adding or removing a single image to the page would completely disrupt the test. Direct creation of JavaScript representations of guidelines instead of macro recording could help avoid some of this, but the definition of guidelines in this tool is highly dependent on a document's particular structure. A more generic way of expressing guidelines that depends, for example, on an object's resultant position on the screen rather than its position in the document structure could be expected.

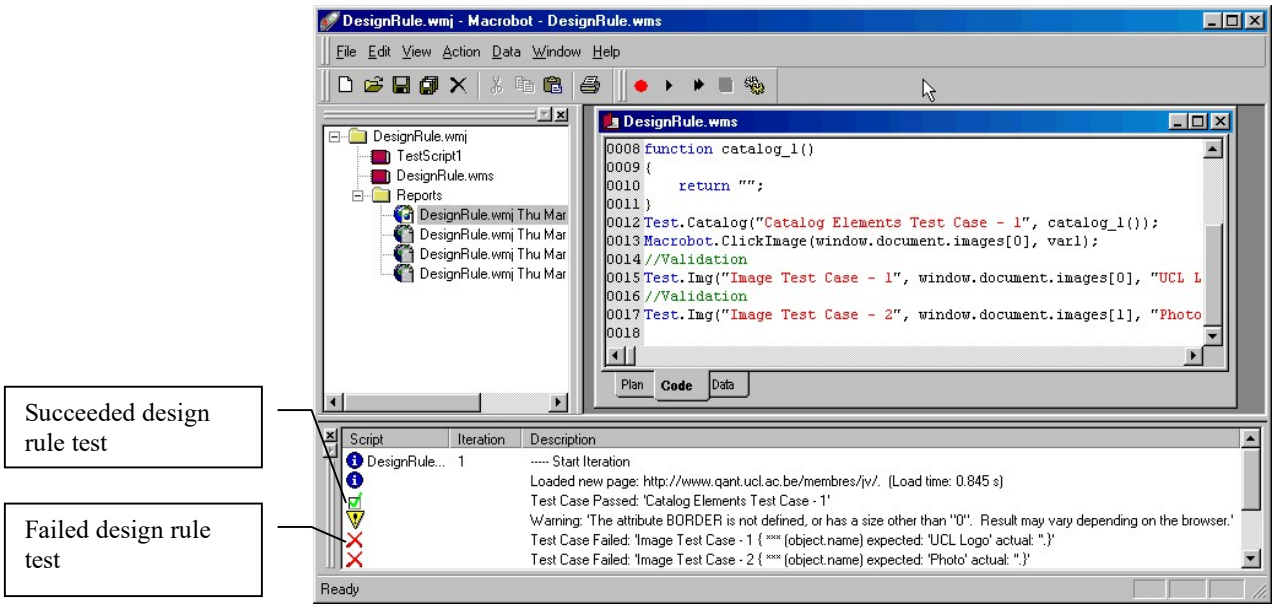

Fig. 10. Replaying a test script for two design rules.

4.2 A guideline from a compilation

Let us consider now a guideline extracted from a guidelines compilation

☞ Select colors that will make your page easy to read by people with color blindness. One good test is to see if your page is readable in back and white (Vanderheiden et al., 96).

This guideline is a typical one that can be used in requirements engineering (step ❶ in fig. 9), in site specification (step ❷ in fig. 9), and in site evaluation (step ❺ in fig. 9). According to the framework, this guideline could be formalized as follows:

1. Requirements engineering
   1.2 Users
      1.2.1.5 Handicaps: impairment = color blindness
2. Site specification
   2.1.1. Spec. C.E.-Compatibility
      2.1.1.2.6 Accessibility: user limitation = lack of color differentiation
      2.1.2.2.8 Colors: colors = readable by users
5. Site evaluation
      5.1.3.4 Color scheme = readable by users; legible in black & white

This guideline, as it is represented, cannot be automated in a straightforward manner. However, if we refer to the research conducted by Murch (Murch,87), a restriction of this guideline can be expressed as: the combination between the background color and the foreground color should belong to the best color combination or should not belong to the worst color combinations. Fig. 11 represents the best color combinations for thin lines and text. It can be read as follows: for thin lines and text displayed on a white background, the best color is blue in more or less 94% of cases, then black in 63% of cases, and finally red with 25% of cases (Murch's research is based on legibility tests). For thin lines and text displayed on a black background, the best color is white in 75% of cases followed by yellow in 63% of cases. The other lines can be interpreted similarly.

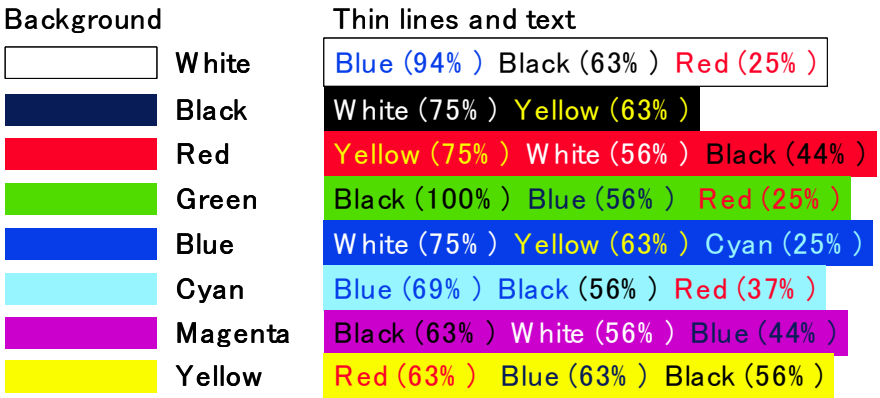

Fig. 11. The best color combinations for thin lines and text.

Fig. 12 represents the best color combinations for bold lines and panels. It can be read as follows: for bold lines and panels displayed on a white background, the best color is black in more or less 69% of cases, then blue in 63% of cases, and finally red with 31% of cases. We consequently observe that usable color combinations also depend of the text style or area. While identifying bold text is easy thanks to the <B> **bold text** </B> tag, identifying bold lines and panels remains more challenging.

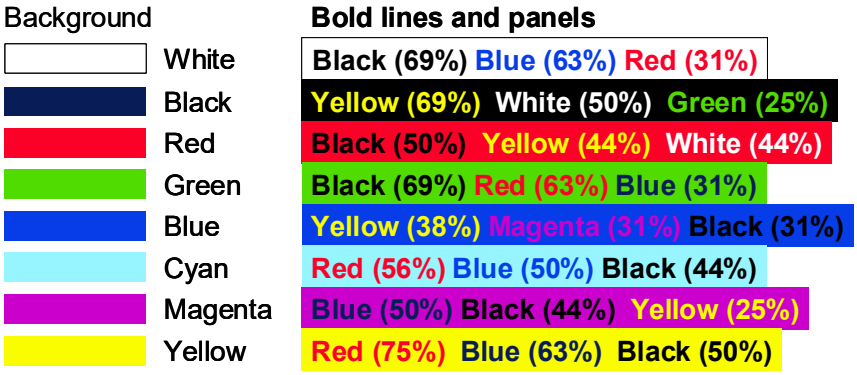

Fig. 12. The best color combinations for bold lines and panels.

Fig. 13 represents the worst color combinations for thin lines and text. It can be read as follows: for thin lines and text displayed on a white background, yellow induces a legibility problem in almost all cases, while Cyan does it in 94% of cases, and so forth.

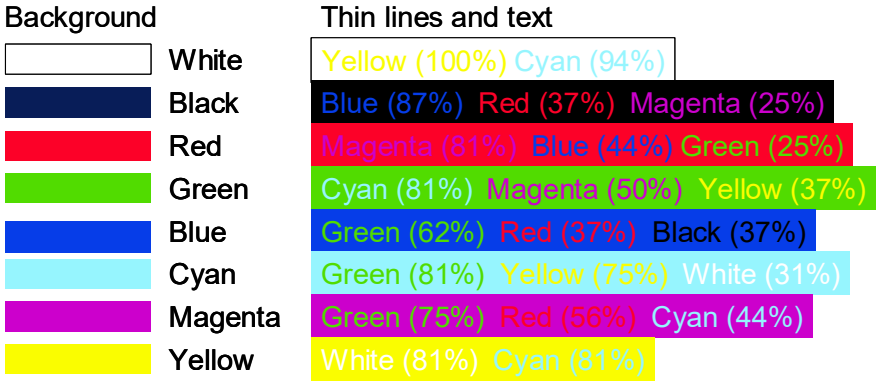

Fig. 13. The worst color combinations for thin lines and text.

Fig. 14 represents the worst color combinations for bold lines and panels. It can be read as follows: for bold lines and panels displayed on a white background, yellow induces a legibility problem in nearly 95% of cases, while Cyan does it in 75% of cases, and so forth.

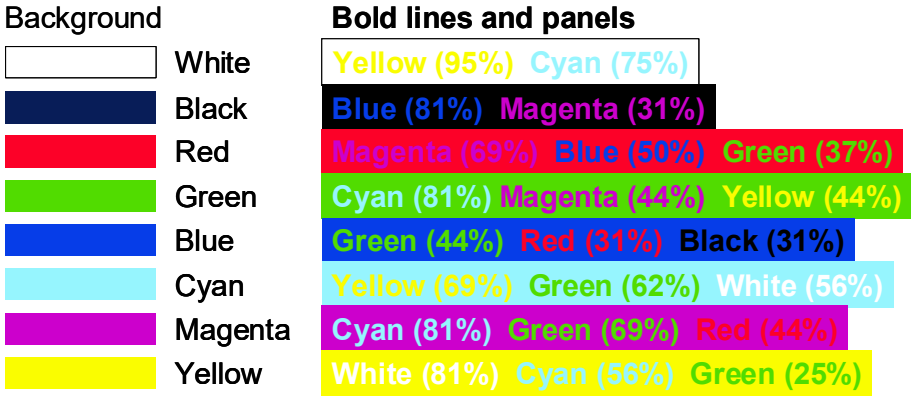

Fig. 14. The best color combinations for bold lines and panels.

Now, having these experimental results in mind, we could more easily think how to automate the testing of this guideline. The different colors used in the HTML code should be identified for this purpose. Color is rendered on computer-based screens as an additive mixture of three primary colors: red, green, and blue, the intensity of which is indicated by a value ranging from 0 to 255. Thus, in the Red-Green-Blue (RGB) model, each color consists in a triple (red-intensity, green-intensity, blue-intensity). This triple is represented in HTML as a hexadecimal code of the type #XXYYZZ, where XX,YY, and ZZ represents the different intensities respectively. Therefore, these codes need to be transformed into decimal for comparison. When these values belong to the set {00,33,66,99,CC, FF}, the resulting colors are said to be principal as no other palette than the basic palette should be loaded to display the colors. The bgcolor and color HTML tags specify the foreground and the background text colors respectively. The resulting procedure for this automated testing is the following:

```
ListOfBestColors1   := {see figure 11};  /* Definition of best color combinations for thin lines and text */
ListOfBestColors2   := {see figure 12};  /* Definition of best color combinations for bold lines and panels */
ListOfWorstColors1 := {see figure 13};  /* Definition of worst color combinations for thin lines and text */
ListOfWorstColors2 := {see figure 14};  /* Definition of worst color combinations for bold lines and panels */
ListOfPages = {URL1, URL2,…, URLn} /* List of the URLs of the web pages to be tested */
for i=1 to # (ListOfPages)                    /* for each web page to be tested */
  ListOfColorComb := DetectPresence ( ListOfPages(i), ColorCombinations )
    /* Detect the presence of multiple color combinations if any; for example, a same color background can be
       used, while the foreground color may be varying within a same page. Each element of the list is character-
       ized by a triple (fgbolor, bgcolor, bold) */
  if ListOfColorComb <> ∅ then              /* if there is at least one color combination detected in the page */
    for k=1 to #(ListOfColorComb)
      CurrentComb.fgcolor := HEX2DEC (ListOfColorComb (k).fgcolor);
       /* Transform the hexadecimal code of the foreground color into decimal */
      CurrentComb.bgcolor := HEX2DEC (ListOfColorComb (k).bgcolor);
      /* Same for background color */
      CurrentComb.bold := ListOfColorComb (k).bold;
      /* Specifies if the bold tag has been detected at the same time */
      if CurrentComb.bold then
        if CurrentComb in ListOfWorstColors2 then /* if the current combination belongs to the worst list… */
          AddUsabPblm (ListOfProblems (ListOfPages (i)), ColorCombPblm, 1, CurrentComb)
              /* Add to the list of usability problems detected for the i-th page the problem identified by
                 ColorCombPblm, which is a reserved constant indicating the type of problem, with the highest
```

```
            severity level (1) for the CurrentComb color combination */
        elseif CurrentComb not in ListOfBestColors2 then /* If the combination does not belong to any lists */
          AddUsabPblm (ListOfProblems (ListOfPages (i)), ColorCombPblm, 3, CurrentComb)
           /* Add to the list of usability problems detected for the i-th page the problem identified by
              ColorCombPblm, which is a reserved constant indicating the type of problem, with the moderate
              severity level (3) for the CurrentComb color combination */
        endif
      else
        if CurrentComb in ListOfWorstColors1 then /* if the current combination belongs to the worst list… */
        AddUsabPblm (ListOfProblems (ListOfPages (i)), ColorCombPblm, 1, CurrentComb)
        elseif CurrentComb not in ListOfBestColors1 then /* If the combination does not belong to any lists */
          AddUsabPblm (ListOfProblems (ListOfPages (i)), ColorCombPblm, 3, CurrentComb)
        endif
      endif
    endfor
  endif
endfor
```

This is a basic procedure which does not support the evaluation of hue, saturation and lightness: each composite color is reduced to its corresponding primary color, thus introducing a restriction of the initial guideline. Graphic backgrounds are not supported although the different colors used in GIF or JPEG files could be identified. But in this case, a spatial algorithm should take care of physical appearance of text on top of graphical background colors, which is far more complex. As we can see here, the cost of developing the complete procedure for automated testing of a single guideline can be very high. However, this implementation being done, the guideline can be tested rapidly.

4.3 A style guide guideline

Style guide guidelines are important to check adherence to some corporate “Look & Feel” guidelines. Style guide compliance increases usability whereas violation will normally decrease it. Let us consider the following style guide example:

- ☞ All data fields must be accessible via a tabulation key in the same order. If fields are grouped together, then tab must access the fields within the group (Levine,96).

Fig. 15 shows an window where this guideline is not addressed: instead of relying on the tab order required by the task, it is relying on the order generated by the system. The different figures indicate the order according to which the different widgets are read or manipulated.

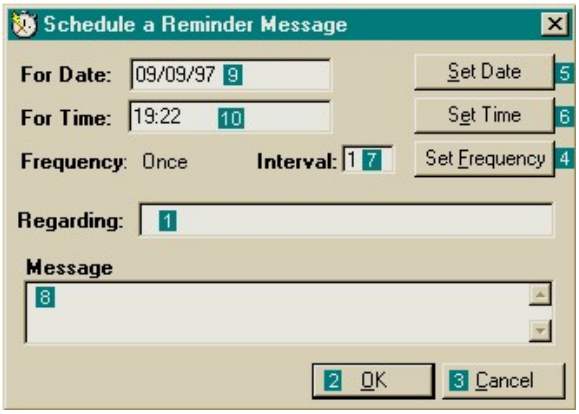

Fig. 15. Disturbing order of fields (Source: http://www.iarchitect.com/controls.htm).

According to the framework, this guideline could be formalized as follows:
1. Requirements engineering
  1.1.1.2 Task order/sequencing: task order = domain objects sequence
2. Site specification
  2.1.1 Spec. C.E.-Compatibility
    2.1.1.1.1.2 Task.order/sequencing: fields order = task order

A procedure to automatically test this guideline would certainly exploit the appearance order of widgets in the HTML code and check whether this order is compatible (hence, the compatibility ergonomic criteria above) with the order recommended by the task. For this purpose, the test can only be achieved if such an order can be provided either manually (e.g., at run-time) or automatically (e.g., from a task model declaratively maintained).

4.4 A standard guideline

Let us consider the following guideline re-expressed from the ISO 9241-Part 12 (ISO,99b) standard:

☞ Error messages should not use humor.

According to the framework, this guideline could be formalized as follows:
3.1.10 Error information: information $\not\subset$ no humor
3.2.2 Conc. C.E.-Quality-ErrorMessages: message $\not\subset$ humor words and expressions

Evaluating this guideline on web pages automatically is quite complex since identifying error messages on web pages is hazardous; testing if a text contains humor is even more hazardous as this is language-, context-, and user-dependent. To solve these problems, a possible restriction of the above guideline could be to submit the text of each web page generated by HTTP error codes (e.g., HTTP 404 File not found) to a detection of commonly used humorous words and expressions. In this case, the original guideline cannot be tested automatically, but its restriction is. The expressiveness and the completeness of the initial guideline have consequently been reduced in this transformation. Hence, it is important to warn the evaluator of this possible loss of expressiveness. It is indeed almost impossible to encompass all humorous words and expressions in a list to be tested for occurrence, even if this list is made user expandable. The resulting procedure for this automated testing is the following:

```
HumorWords = {blablabla, aha, hmm, …}  /* To be enriched as experience is growing */
Humor Expressions = {joke1,joke2,…}
ListOfPages = {URL1, URL2,…, URLn} /* List of the URLs of the web pages to be tested */
for i=1 to # (ListOfPages)                    /* for each web page to be tested */
  ListOfMsg := DetectPresence ( ListOfPages(i), ErrorMessages )
     /* Detects the presence of error messages produced by HTTP protocol and returns the results in a list */
  if ListOfMsg <> ∅ then                      /* if there is at least one error message detected in the page */
    for k=1 to #(ListOfMsg)
      if IsThereHumWords ( ListOfMsg(k), WordIndex, HumorWords) or
        /* Returns 'true' if a regular expression belonging to the list HumorExpressions is found in ListOfMsg(k) */
        IsThereHumExp ( ListOfMsg(k), WordIndex, HumorExpressions)
        /* Returns 'true' if a word belonging to the list HumorWords is found in ListOfMsg(k) */
        then AddUsabPblm (ListOfProblems (ListOfPages (i)), HumMsgPblm, WordIndex, ListOfMsg(k))
              /* Add to the list of usability problems detected for the i-th page the problem identified by
                 HumMsgPblm, which is a reserved constant indicating the type of problem, at location WordIndex
                 for the k-th message in the list */
      endif
    endfor
  endif
endfor
```

4.5 Guidelines in an ergonomic algorithm

Let us consider the following guideline re-expressed from an ergonomic algorithm (Kim & Foley,93):

☞ Links should be consistent with related page titles.

According to the framework, this guideline could be formalized as follows:

2.1.2.5 Spec. C.E.-Guidance
  2.1.2.5.1.3.2 Identification/Title
    2.1.2.5.1.3.2.1 Windows: title = name of task
3.1 Conc. C.E.-Consistency
  3.1.5. Labels
    3.1.5.1 Links: label = title

Two situations for testing this guideline may occur: either the link label is exactly the related page title or at least one difference exists. The automation is straightforward in the first situation, while impossible in the second: from a semantic point of view, it is impossible to prove that a link label is semantically consistent with the related page title. However, a similarity score used in Information Retrieval can be computed from the amount of shared words and shared ordering: if the two expressions share a significant amount of terms in a relatively similar ordering, then it can be assumed that a syntactical consistency is established. Again, the resulting guideline is a restriction to the syntactic domain. The resulting procedure for this automated testing is the following:

```
Threshold := constant;                    /* Definition of a certain threshold beyond which it is considered that
                                             two different expressions are not syntactically related with respect
                                             to the Similarity Factor */
ListOfPages = {URL1, URL2,…, URLn} /* List of the URLs of the web pages to be tested */
for i=1 to # (ListOfPages)                /* for each web page to be tested */
  ListOfLinks := DetectPresence ( ListOfPages(i), Links )
    /* Detects the presence of links in the i-th page and returns their names in ListOfLinks */
  if ListOfLinks <> ∅                     /* if there is at least one link detected in the page */
   then
    for j=1 to #(ListOfLinks)
      TargetTitle := DetectPresence ( ListOfLinks (j), Title)
                    /* Returns the <TITLE> tag of the related web page if any */
      if TargetTitle = ∅
        then /* The target title is missing: another potential usability problem can de detected here */
          AddUsabPblm (ListOfProblems (ListOfPages (i)), MisTitlePblm, null, null)
              /* Add to the list of usability problems detected for the i-th page the problem identified by
                 MisTitlePblm, which is a reserved constant indicating the type of problem */
        else /* The target title exists */
          SimilarityFactor := ComputeSimilarity (Name(ListOfLinks(j)), TargetTitle)
             /* Compute the similarity factor used in information retrieval between the two expressions:
                  the name of the link being currently used and the title of the related page */
          if SimilarityFactor <= Threshold  /* If this factor is not important enough, i.e. beyond a certain threshold */
            then AddUsabPblm (ListOfProblems (ListOfPages (i)), InconsTitlePblm, null, Name(ListOfLinks(j)))
              /* Add to the list of usability problems detected for the i-th page the problem identified by
                 InconsTitlePblm, which is a reserved constant indicating the type of problem, for the name of the
                 j-th link in this page */
          endif
      endif
    endif
endfor
```

5. Related work

Automatic Usability Software (AUS) consists in a client-installed system that monitors and record data on user actions as they are carrying out their interactive tasks (Chang & Dillon,97). Such data include for example the number of user request from or to the server, the mouse movements, keystrokes, the density of mouse clicks. It is similar to Wet (Etgen & Cantor,99) as it captures data on the client machine, but Wet is focusing more on user events to be sent to the server for analysis. Evaluators are able to copy/paste data files from any client workstation to their analysis system, provided that test users are carrying out the same tasks. AUS is also equipped with a playback facility that replays user actions from their definition in the files, such as in MacroBot (Fig. 10), AUS is claimed to be a tool for automated web site usability, but it is more intended to automate data capture rather than its evaluation, as no assistance is provided on how to process and analyze them (except an overall figure of merit). However, some data captured by AUS could easily **inform** the left-hand side of guidelines in EvalWeb.

Bauer & Scharl introduced a framework for assessing web sites divided into five classification criteria (i.e., content, interactivity, design, web-site management, and security), which are in turn decomposed into sub-criteria (Bauer & Scharl,99). A robot automatically examines web sites to capture the value of some of the (sub-)criteria, thus avoiding repetitive and error-prone human activities. While some (sub-)criteria obviously affect the usability (e.g., the navigational structure is a nominal sub-criteria measured from the use of frames), there is no true usability evaluation of web sites. As for AUS, information gathered in this framework may inform some EvalWeb guideline.

Usability Testing and Analysis Facility (UTAF) is software for automatic interface inspection intended to automatically test high-risk software interface against NASA usability guidelines (Whitmore & Berman). It is highly related to Computer-Human Interaction Models (CHIMES), which is aimed at automatically test usability guidelines related to static interface properties such as button size, labeling, location, fonts, point size, and the use of colors (Jiang et al., 92a,b). These guidelines come from many sources: style guides (e.g., OSF/Motif style guide), compilation of guidelines (e.g., the Smith & Mosier document, Mayhew's book), and standard (e.g., the MIL-STD-1472D standard). CHIMES can accommodate both character-based user interface (CUI) and graphical user interfaces (GUI), but not web pages. An interesting feature of CHIMES concerns its ability to provide advice and comments why a particular guidelines is violated on a certain object. It does this by enabling the designer to request such advice in the context of the GUI being tested.

KRI/AG automatically tests presentation guidelines (Löwgren & Nordqvist,92) from standards and OSF/Motif style guide for a Motif user interface whose description in captured in User Interface Language (UIL). It is very similar to CHIMES in this aspect. Designers who tested KRI/AG complained namely that users and task characteristics are not considered in the usability testing because guidelines related to these aspects do not have access to relevant information or because they are merely to difficult to interpret or to implement (Löwgren & Laurén, 93). To overcome this shortcoming, our framework is providing an explicit way of describing these information needs, thus clarifying the interpretation. As this information cannot be retrieved from HTML code, other sources might be investigated. Data contained in web log files appears the first eligible candidate.

But at second glance, these log files are more intended to measure the **utilization** of the web pages rather than their **usability**. The *utilization* refers to the current usage of the web site by users, whether the

web site is usable or not. Conversely, *usability* refers to the adequacy on the current utilization of a web site with respect to the context of use, including cognitive capabilities and preferences of users. In isolation, utilization metrics do not enable us to easily detect usability problems. But when these are related to values provided by usability guidelines, some precise diagnostic can be produced. For instance, the most frequently used path to information does not mean a lot in itself. But if the average time required by this path is known and compared to a reference value (e.g., which can be computed by action analysis, cognitive modeling, or predictive models; Kieras, Scott & Meyer,97), any significant deviation between the current utilization (which is user observed) and the expected utilization (which is supposed to be usable) may predict the presence of a usability problem.

Sherlock (Mahajan & Shneiderman, 96) automatically checks some individual usability guidelines and consistency guidelines across dialog boxes of Visual Basic user interfaces. It is capable of identifying variations in style, font, colors, in labels used in dialog boxes (e.g., "quit" vs. "Quit"), in buttons labels and order (e.g., Ok → Cancel vs. Cancel → Ok), in words not belonging to a dictionary, in words used for referencing to the same task, but in different dialog boxes. These guidelines are probably the most straightforwardly represented within our framework since it mainly invokes DetectPresence and Compare functions. But these guidelines provide already an extensive set of ideas to test.

Nevertheless, log files can be combined with other models, like user, dialog, and task models: this family of solutions is best known as *model-based approaches*. In these methods, data could be collected from users as they carry out their interactive task and compared with the reference model.

RemUSINE is considering this approach, but does not embed guidelines: only a comparison between log files and a task models (Lecere & Paternò,98). But usability aspects can be automated, for example, one can deduce some results from the most frequently used pages path, errors, help requests, and so on (fig. 16). Provided results can be pertained to a single user or a group of users, but also for task-related aspects. For instance, "the task model supported by the user interface imposes temporal constraints [that are] too rigid for the conceptual model of the user." (Paternò & Ballardin, 99)

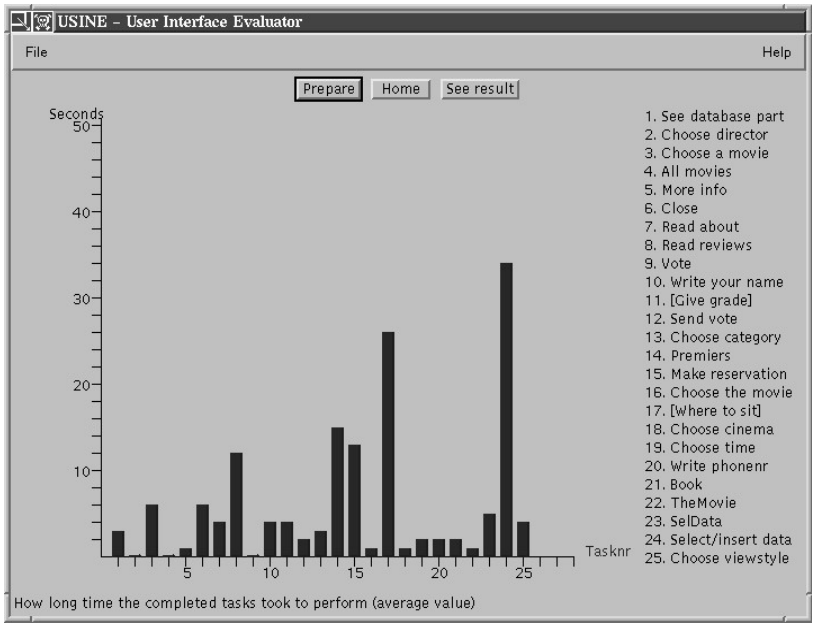

Fig. 16. Results of RemUSINE tool.

USAGE (Byrne et al.,94) automatically analyzes user actions captured in log files obtained when users are carrying out their interactive tasks according to a GOMS approach (Kieras et al.,95).

ÉMA (Balbo,95) is an automatic monitoring tool that captures users' actions, too, compare them with a data-flow representation as dialog model of the interface being tested and automatically detects common behavioral patterns based on dialog guidelines. It consequently produces an analysis helping the designer to identify potential dialog usability problems. This work can substantially increase our dialog guidelines, which are more complex to interpret and thus to test than presentation guidelines.

6. Conclusion and Future Work

Our conclusion plaids for automated testing of web usability guidelines with an explicit definition of guidelines, with access to log files, and access to both user and task models. But it would be a tremendous challenge to integrate all these into a single piece of software. Therefore, progress can be imagined independently along several dimensions:

- *Dimension of guidelines*: a complete and consistent definition for a Guideline Definition Language (GDL) is expected to uniformly represent the information they involve and to communicate how to apply them. Several guidelines can be automated while some other cannot. A wider taxonomy of the reasons why they can or cannot be automated is needed. Finally, an important future work would be to examine each EvalWeb guideline, to list information needed to evaluate them (manually or automatically) in GDL terms, to identify where in these tools and methods these data can be extracted, and to incorporate them in the evaluation process. Of course, at the moment, not all guidelines, whatever their type is, can be tested automatically without any human control and intelligence. Especially until now, we have not yet fully formalized all existing guidelines nor check enough guidelines. They are increasing every day. On the other hand, guidelines that cannot be formalized (even after some framework adaptation) are likely to be non-testable and therefore should be considered instead as high-level principles.
- *Dimension of the heuristic evaluation method*: usability testing can reveal usability flaws in poorly designed pages but cannot prove that a particular web page is well executed. This is a rather negative approach: if some usability problems are detected, then the corresponding pages are considered not usable and their lack of usability is proportional to the number of flaws revealed in the testing. If no flaws are detected against the guidelines, one can be sure that they guarantee some usability, but not the complete desired usability. Moreover, it is likely that some other flaws remain unrevealed because no existing guidelines are addressing them. Finally, for some very original, unique web sites (e.g., www.chman.com), little or no guidelines can be used to prove their original, yet usable, design. A more positive approach can be desired here.
- *Dimension of involved models*: designers in the study conducted by Löwgren and Laurén (1993) reported that they wanted to customize the results of the critique to their needs, to have not always the same repeatedly message produced, to choose different levels of severity, and to have more control on the evaluation process. A significant contribution here would be to examine how these concerns can be addressed in the supporting tool and test if a benefit is resulting from it.
- *Dimension of supporting tools*: this study identified some of the potential benefits and some shortcomings of automated testing of usability guidelines. A prototype is beginning to be implemented; but based on these preliminary results, it appears that a more fully developed version of the tool would be a promising contribution to automated testing of usability guidelines.

Acknowledgements

This work has been carried out under the auspices of EvalWeb research project (http://lis.univ-tlse1.fr/evalweb/), funded by research program "Cognition et Conception", G.I.S. Sciences de la Cognition (http://www-poleia.lip6.fr/gis.cognition/), of the Centre National de la Recherche Scientifique (CNRS), the MetroWeb research project funded by Région Wallonne and by a Tournesol mobility program. It is also an initiative from the International Special Interest Group (SIG) on Tools for Working with Guidelines.

References

1. Balbo, S., *Software Tools for Evaluating the Usability of User Interfaces*, in Proceedings of the 6th International Conference on Human-Computer Interaction HCI International'95 (Yokohama, July 9-14, 1995), Elsevier, Amsterdam, 1995, pp. 337–342.
2. Bastien, J.M.C. and Scapin, D.L., *Evaluating a User Interface with Ergonomic Criteria,* International Journal of Human-Computer Interaction, Vol. 7, 1995, pp. 105–121.
3. Bastien, J.M.C., Scapin, D.L., Leulier, C., *The Ergonomic Criteria and the ISO 9241-10 Dialogue Principles: A comparison in an evaluation task*, Interacting with Computers, 11(3), 1999, pp. 299–322.
4. Bauer, Ch., Scharl, A., *A Classification Framework and Assessment Model for Automated Web Site Evaluation*, in Proceedings of 7th European Conference on Information Systems (Copenhagen, June 23-25, 1999), J. Pries-Heje, C. Liborra, K. Kautz, J. Valor, E. Christiaanse, D. Avison, C. Heje (eds.), Vol. III, 1999, pp. 758–765.
5. Borges, J.A., Morales, I. and Rodríguez, N.J., *Guidelines for Designing Usable World Wide Web Pages,* in Companion Proceedings of ACM Conference on Human Aspects in Computing Systems CHI'96 (Vancouver, April 14-18, 1996), ACM Press, New York, 1996, pp. 277–278. Accessible at http://www.acm.org/sigchi/chi96/proceedings/shortpap/Rodriguez/rn_txt.htm
6. Bowers, N., *Weblint: Quality Assurance for World-Wide Web*, in Proc. of 5th Int. World Wide Web Conf. WWW'5 (Paris, May 6-10, 1996), Elsevier, Amsterdam, 1996. Accessible at http://www5conf.inria.fr/fich_html/papers/P34/Overview.html
7. Byrne, M.D., Wood, S.D., Sukaviriya, P., Foley J.D. and Kieras, D., *Automating Interface Evaluation Automatic Support in Design and Use*, in Proc. of CHI'94 (Boston, April 24-28, 1994), ACM Press, New York, 1994, pp. 232–237. Accessible at http://www.acm.org/pubs/articles/proceedings/chi/191666/p232-byrne/p232-byrne.pdf
8. Chang, E. and Dillon, T.S., *Automated Usability Testing*, in Proceedings of IFIP TC. 13 International Conference on Human-Computer Interaction Interact'97 (Sydney, June 1997), S. Howard, J. Hammond, G. Lindgaard (eds.), Chapman & Hall, London, 1997, pp. 77–84.
9. Clark, D. and Dardailler, D., *Accessibility on the Web: Evaluation & Repair tools to make it possible*, 1999. Accessible at http://www.dinf.org/csun_99/session0195.html
10. Comber, T., *Building Usable Web Pages: An HCI Perspective*, in Proceedings of Australian Conf. on the Web AusWeb'95 (Ballina, August 1995), Nordsearch Ltd, Ballina, 1995, pp. 119–124.
11. Cooper, M. *Evaluating Accessibility and Usability of Web Pages*, in Proc. of 3rd International Conference on Computer-Aided Design of User Interfaces CADUI'99 (Louvain-la-Neuve, October 21-23, 1999), Kluwer Academics, Dordrecht, 1999, pp. 33–42.
12. M. Cooper, *Universal Design of a Web Site*, in Proceedings of Technology and Persons with Disabilities CSUN'99 (Los Angeles, March 15-20, 1999), California State University Northridge, Center on Disabilities (COD), 1999. Accessible at http://www.dinf.org/csun_99/session0030.html
13. Detweiler, M.C and Omanson, R.C., *Ameritech Web Page User Interface Standards and Design Guidelines*, Ameritech Corp., Chicago, 1996. Accessible at http://www.ameritech.com/corporate/testtown/library/standard/web_guidelines/index.html
14. Etgen, M. and Cantor, J., *What does getting WET (Web Event-logging Tool) Mean for Web Usability?*, in Proceedings of the 5th International Conference on Human Factors and the Web HFWeb'99 (Gaithersburg, June 3, 1999), S. Laskowski (ed.), 1999. Accessible at http://zing.ncsl.nist.gov/hfweb/proceedings/etgen-cantor/index.html
15. Faraday, P., *Visually Critiquing Web Pages*, in Proc. of IFIP TC. 13 Conf. on Human-Computer Interaction INTERACT'99 (Edinburgh, August 1999), A. Sasse & Ch. Johnson (eds.), IOS Press, 1999.
16. Farenc, Ch., Liberati, V. and Barthet, M.-F., *Automatic Ergonomic Evaluation: What are the Limits?*, in Proceedings of 2nd International Conference on Computer-Aided Design of User Interfaces CADUI'96 (Namur, June 5-7, 1996), J.

Vanderdonckt (ed.), Presses Universitaires de Namur, Namur, 1996, pp. 159–170. Accessible at http://lis.univ-tlse1.fr/~farenc/papers/cadui 96-cf.ps

17. Grose, E.M., Forsythe, Ch., Ratner, J. (eds.), *Human Factors and Web Development*, Lawrence Erlbaum Associates, Mahwah, 1998.
18. Harrison, M.D., Thimbleby, H.W., *Formalising Guidelines for the Design of Interactive Systems*, in Proceedings of the HCI'85 Conference on People and Computers (University of East Anglia, September 17-20, 1985), P. Johnson & S. Cook (eds.), Cambridge University Press, Cambridge, 1985, pp.161–171.
19. Hartson, R., Castillo, J., Kelso, J., Kamler, J., and Neale, W., *Remote Evualation: The Network as an Extension of the Usability Laboratory*, in Proc. of ACM Conference of Human Aspects in Computing Systems CHI'96 (Vancouver, April 14-18, 1996), ACM Press, New York, 1996, pp. 228–235. Accessible at http://www.acm.org/sigchi/chi96/proceedings/papers/Hartson/hrh_txt.htm
20. HFES/ANSI 200 Standard: Draft, Human Factors & Ergonomics Society, Santa Monica, 1997.
21. ISO Draft International Standard (DIS) 9241-11, *Ergonomic Requirements for office work with visual display terminals, Part 11: Guidance on Usability*, International Standards Organization, Geneva, 1999.
22. ISO Draft International Standard (DIS) 9241-12, *Ergonomic Requirements for office work with visual display terminals, Part 12: Presentation of Information*, International Standards Organization, Geneva, 1999.
23. Jiang, J., Murphy, E.D., Bailin, S.C., Truszkowski, W.F., *Automating a human factors Evaluation of Graphical User Interfaces for NASA Applications : An update on CHIMES*, in Proceedings of the 2nd International Symposium on Ground Data Systems for Space Mission Operations SpaceOps'92, JPL Publication 93-5, Pasadena, Jet Propulsion Laboratory, 1992, pp. 685–690.
24. Jiang, J., Murphy, E.D., Bailin, S.C., Truszkowski, W.F., Szczur, M.R., *Prototyping a Knowledge-based compliance checker for User-Interfac Evaluation on Motif development environments*, in Proc. of 2nd Annual International Motif Users Meeting Motif'92 (Bethesda, 1992), Open Systems, 1992, pp. 258–268.
25. Kieras, D.E., Scott, W., Meyer, D., *Predictive Engineering Models Based on the EPIC Architecture for a Multimodal High-Performance Human-Computer Interaction Task*, ACM Transactions on Computer-Human Interaction, Vol. 4, No. 3, September 1997, pp. 230–275.
26. Kieras, D.E., Wood, A.D., Abotel, K., Hornof, A., *GLEAN: A Computer-Based Tool for Rapid GOMS Model, Usability Evaluation of User Interface Designs*, in Proceedings of ACM Conf. on User Interface Software Technology UIST'95, ACM Press, New York, 1995, pp. 91–100.
27. Kim, W.C., J.D. Foley, *Providing High-Level Control and Expert Assistance in the User Interface Presentation Design,* in Proceedings of ACM Conference on Human Factors in Computing Systems INTERCHI'93 (Amsterdam, 24-29 April 1993), ACM Press, New York, 1993, pp.430–437.
28. Lecerof, A. and Paternò, F., *Automatic Support for Usability Evaluation*, IEEE Transactions of Software Engineering, Vol. 24, No. 10, October 1998, pp. 863–888.
29. Levine, R.: User Interface Design for Sun Microsystem's Internal Web. Sun Microsystems Inc. (1996). Accessible at http://www.sun.com/styleguide
30. Linz, T. and Daigl, M., *How to Automate Testing of Graphical User Interfaces*, ESSI PIE 24306 research report, imbus GmbJ, Möhrendorf, 1998. Accessible at http://www.imbus.de/html/GUI/AQUIS-full_paper-1.3.html
31. Löwgren, J. and Nordqvist, T., *Knowledge-Based Evaluation as Design Support for Graphical User Interfaces*, in Proc. of ACM Conference on Human Aspects in Computing Systems CHI'92 (Monterey, May 3-7, 1992), ACM Press, New York, 1992, pp. 181–188. Accessible at http://www.hcibib.org/gs.cgi?word=checked&terms=C.CHI.92.181
32. Löwgren, J., Laurén, U., *Supporting the use of guidelines and style guides in professional user interface design,* Interacting With Computers, Vol. 5, 1993, pp. 385-396.
33. Lynch, G., Palmiter, S. and Tilt, Ch., *The Max Model: A Standard Web Site User Model*, in Proceedings of the 4th International Conference on Human Factors and the Web HFWeb'98 (Basking Ridge, June 5, 1998), J. Cantor (ed.), 1998. Accessible at http://zing.ncsl.nist.gov/hfweb/proceedings/lynch/index.html
34. Mahajan, R., Shneiderman, B., *Visual & textual consistency checking tools for graphical user interfaces*, Technical Report CS-TR-3639, University of Maryland, 1996.
35. Mayhew, D., *Principles and Guidelines in Software User Interface Design*, Prentice Hall, Englewood Cliffs, 1992.
36. Murch, G.M., *Colour Graphics - Blessing or Ballyhoo?*, Chapter 7: The Visual Channel, in R.M. Baecker & W.A.S. Buxton (eds.), Readings in Human-Computer Interaction - A Multidisciplinary Approach, Morgan Kaufmann Publishers, Los Altos, 1987, pp. 333–341.

37. Nielsen, J., *Enhancing the Explanatory Power of Usability Heuristics Tools for Design*, in Proceedings of ACM Conference on Human Factors in Computing Systems CHI'94 (Boston, 24-28 April 1994), ACM Press, New York, 1994, pp.152–158.
38. Nielsen, J., Mack, R.M., *Usability Inspection Methods*, John Wiley & Sons, New York, 1994.
39. Paternò, F., Ballardin, G., *Model-aided Remote Usability Evaluation*, in Proc. of IFIP TC. 13 Conf. on Human-Computer Interaction INTERACT'99 (Edinburgh, August 1999), A. Sasse & Ch. Johnson (eds.), IOS Press, 1999, pp. 434-442.
40. Ratner, J., Grose, E.M., Forsythe, Ch., *Characterization and Assessment of HTML Style Guides*, in Proceedings of ACM Conference on Human Factors in Computing Systems CHI'96 (Vancouver, 14-18 April 1996), ACM Press, New York, 1996, pp.115–116. Accessible at http://www.acm.org/sigchi/chi96/proceedings/intpost/Ratner/rj_txt.htm
41. Scapin, D.L., Bastien, J.M.C., *Ergonomic criteria for evaluating the ergonomic quality of interactive systems*, Behaviour & Information Technology, 16, 1997, pp. 220–231.
42. Scapin, D.L., Guarrigues, S., Farenc, C., Vanderdonckt, J., Palanque, P., Bastide, R., Bastien, J.M.C. and Leulier, C., *Conception ergonomique d'interfaces web: démarche et outil logiciel de guidage et de support*, Rapport final du projet EvalWeb, Appel d'offre "Cognition et Conception", GIS Sciences de la Cognition, CNRS, Paris-Toulouse-Louvain-la-Neuve, December 1999.
43. Scholtz, J., Laskowski, S. and Downey, L., *Developing Usability Tools and Techniques for Designing and Testing Web Sites*, in Proceedings of the 4th International Conference on Human Factors and the Web HFWeb'98 (Basking Ridge, June 5, 1998), J. Cantor (ed.), 1998. Accessible at http://www.research.att.com/conf/hfweb/proceedings/scholtz/index.html
44. Scholtz, J., *WebMetrics: A Methodology for Producing Usable Web Sites*, in Proceedings of the 42nd Annual Meeting of Human Factors and Ergonomics Society HFES'98 (Chicago, October 5-9, 1998), Vol. E, Human Factors and Ergonomics Society, Santa Monica, pp. 1612.
45. Vanderdonckt, J., *Guide ergonomique des interfaces homme-machine*, Presses Universitaires de Namur, Namur, 1994.
46. Vanderdonckt, J., *Development Milestones towards a Tool for Working with Guidelines*, Interacting with Computers, Vol. 12, N°2, 1999, pp. 81-118. Accessible at http://belchi.qant.ucl.ac.be/publi/1999/Milestones.pdf
47. Vanderheiden, G.C., Chisholm, W.A., Ewers, N., Dunphy, S. M., *Unified Web site accessibility guidelines*, Version 7.2, Trace Center, University of Wisconsin-Madison, 1997. Accessible at http://trace.wisc.edu/text/guidelns/htmlgide/htmlgide.htm
48. *WAI Accessibility Guidelines*, Web Accessibility Initiative, World Wide Web Consortium, Geneva, 1998. Accessible at http://www.w3.org/wai
49. Whitmore, M. and Berman, A.H., *Common Ground for Critical Shuttle and Space Station User Interfaces: An Independent Verification and Validation Approach*, in Proceedings of ACM Conference on Human Factors in Computing Systems CHI 96 (Vancouver, April 14-18, 1996), ACM Press, New York, 1996, vol. 2, pp.73-74.
50. Yahoo category on HTML Validation Checkers, 2000. Accessible at http://dir.yahoo.com/Computers_and_Internet/Information_and_Documentation/Data_Formats/HTML/Validation_and_Checkers/
51. *Yale C/AIM Web Style Guide*, Yale-New Haven Medical Center, 1997. Accessible at http://info.med.yale.edu/caim/manual/contents.html

Appendix 1. Formalization framework

1. REQUIREMENTS ENGINEERING

1.1. TASKS

1.1.1. Task Structure

1.1.1.1. Task organization

1.1.1.2. Task order/ Sequencing

1.1.1.3. Task stability

1.1.1.4. Timing

1.1.1.5. Specific tasks

1.1.1.5.1. Visual search

1.1.1.5.2. User attention

1.1.1.5.3. Discrimination 2 categories

1.1.2. Information needed
1.1.2.1. Nature
1.1.2.2. Status
1.1.2.3. Order
1.1.2.4. Groups
1.1.2.5. Localisation
1.1.3. Information about Tasks
1.1.3.1. General
1.1.3.2. General conventions
1.1.3.3. Application domain conventions
1.1.3.4. Language conventions
1.1.3.5. Coding conventions
1.2. USERS
1.2.1. Users characteristics
1.2.1.1. Several Populations w. Differents Skills/ Experience levels
1.2.1.2. Level of experience in the tasks
1.2.1.2.1. Reading level
1.2.1.3. Level of experience with computers
1.2.1.3.1. Unfamiliar with field format ?
1.2.1.4. Cultural characteristics (e.g., language, reading order, etc .)
1.2.1.5. Handicaps
1.3. EQUIPMENT
1.3.1. Compatibility color vs monochrome displays

2. SITE SPECIFICATION

2.1. SPEC. CONTENTS
2.1.1. Spec. C.E.-COMPATIBILITY
2.1.1.1. Spec. C.E.-COMPATIBILITY-Tasks
2.1.1.1.1. Structure Tasks
2.1.1.1.1.1. Task Organization
2.1.1.1.1.2. Task Order/Sequencing
2.1.1.1.1.3. Stability
2.1.1.1.1.4. Timing/ ShortCuts
2.1.1.1.2. Information Needed
2.1.1.1.2.1. Nature
2.1.1.1.2.2. Order
2.1.1.1.2.3. Numbering
2.1.1.1.2.4. Grouping
2.1.1.1.2.5. Localisation
2.1.1.1.2.6. Coding
2.1.1.1.2.7. Formats
2.1.1.1.2.8. Cursors
2.1.1.1.3. Information about Tasks
2.1.1.1.4. Other Task related
2.1.1.2. Spec. C.E.-COMPATIBILITY-USERS
2.1.1.2.1. Several Populations w. Differents Levels of'Experience/ Skills
2.1.1.2.2. Task experience level
2.1.1.2.3. Experience with computers
2.1.1.2.3.1. Inexperienced
2.1.1.2.3.2. Experienced
2.1.1.2.4. Reading levels
2.1.1.2.5. Cultural differences
2.1.1.2.6. Accessibility (User Limitations)

2.1.1.2.7. Spec. C.E.-USERS'EXPERIENCE
2.1.1.2.7.1. Other User related
2.1.1.3. Spec. C.E.-COMPATIBILITE-EQUIPMENT
2.1.2. Spec. Contenu Non Task/User/Equipment-Specific
2.1.2.1. General
2.1.2.1.1. General Spec. Displays
2.1.2.1.2. General Spec. Dialogue & Controls
2.1.2.2. Spec. C.E.-SIGNIFICANCE/CODING
2.1.2.2.1. General
2.1.2.2.2. Uniqueness
2.1.2.2.3. Distinctness
2.1.2.2.4. Coding schemes
2.1.2.2.5. Showing Relationships
2.1.2.2.5.1. Windows (Primary/ Secundary)
2.1.2.2.6. Windows Control
2.1.2.2.7. Abbreviations
2.1.2.2.8. Colors
2.1.2.2.9. Icons
2.1.2.2.10. Geometric Shapes
2.1.2.2.11. Blinking
2.1.2.2.12. Sizes
2.1.2.2.13. Voice
2.1.2.3. Spec. . C.E.-Explicit Control
2.1.2.3.1.1. Spec. . C.E.-EXPLICIT USER ACTION
2.1.2.3.1.2. Spec. . C.E.-USER CONTROL
2.1.2.4. Spec. . C.E.-WORKLOAD
2.1.2.4.1. Spec. . C.E.-BREVITY
2.1.2.4.1.1. Spec. . C.E.-CONCISENESS
2.1.2.4.1.2. Spec. . C.E.-MINIMAL ACTIONS
2.1.2.4.1.2.1. Cursor
2.1.2.4.1.2.2. etc.
2.1.2.4.2. Spec. . C.E.-INFORMATION DENSITY
2.1.2.5. Spec. C.E.-GUIDANCE
2.1.2.5.1. Spec. C.E.-PROMPTING
2.1.2.5.1.1. Contents
2.1.2.5.1.1.1. General
2.1.2.5.1.2. Prompting Semantics
2.1.2.5.1.3. Prompting Syntax
2.1.2.5.1.3.1. Cues
2.1.2.5.1.3.2. Identification/ Title
2.1.2.5.1.3.2.1. Windows
2.1.2.5.1.3.2.2. Icons
2.1.2.5.1.3.2.3. Forms
2.1.2.5.1.3.3. Special Prompting Codes
2.1.2.5.1.3.3.1. Markers
2.1.2.5.1.3.3.2. Line orientation
2.1.2.5.1.3.4. Status
2.1.2.5.1.3.5. Entry Fields
2.1.2.5.1.3.6. Units of Measurements
2.1.2.5.1.3.7. Showing Relationships
2.1.2.5.1.3.7.1. Any Text
2.1.2.5.1.3.7.2. Text Fields

2.1.2.5.1.3.7.3. Lists
2.1.2.5.1.3.7.4. Tables
2.1.2.5.1.3.8. Auditory
2.1.2.5.1.3.9. Cursors
2.1.2.5.1.3.10. Dragging
2.1.2.5.1.3.11. Sizing
2.1.2.5.1.3.12. Rotation
2.1.2.5.2. Spec. C.E.-GROUPING/DISTING
2.1.2.5.2.1. Spec. C.E.-BYLOCATION
2.1.2.5.2.2. Spec. C.E.-BYFORMAT
2.1.2.5.2.2.1. Cursor
2.1.2.5.2.2.2. Windows
2.1.2.6. Conc. C.E.-LEGIBILITY
2.1.2.7. Spec. C.E.-FEEDBACK

3. SITE DESIGN

3.1. CONC. C.E.-CONSISTENCY
3.1.1. Windows Format
3.1.2. Areas to Distinguish
3.1.3. Prompts
3.1.4. Commands & Controls
3.1.5. Labels
3.1.6. Coding (& Messages)
3.1.7. Home position
3.1.8. Markers
3.1.9. Status Information
3.1.10. Error Information
3.1.11. On line Help
3.1.12. Modes
3.1.13. Options Location
3.2. CONC. C.E.-ERROR/MANAGEMENT
3.2.1. Conc. C.E.-ERROR PROTECTION
3.2.2. Conc. C.E.-QUALITY-ErrorMessages
3.2.3. Conc. C.E.-ERROR CORRECTION
3.3. CONC. C.E.-FLEXIBILITY
3.4. CONC. HELP on LINE
3.5. CONC. DOCUMENTATION

4. SITE DEVELOPMENT (to be expanded)

5. SITE USAGE AND EVALUATION (to be expanded)

6. SITE MAINTENANCE (to be expanded)