Automated Evaluation of Menu by Guidelines Review

Sara Bouzit[1,2], Gaëlle Calvary[2], Denis Chêne[1], Jean Vanderdonckt[3]
[1]Orange Labs, 28 chemin du Vieux Chêne, F-38240 Meylan (France)
{sarah.bouzit, denis.chene}@orange.com
[2]Univ. Grenoble Alpes, LIG,
CNRS, LIG, F-38000 Grenoble (France) - {sara.bouzit, gaelle.calvary}@imag.fr
[3]Université catholique de Louvain, Louvain School of Management, Louvain Interaction Lab.
B-1348 Louvain-la-Neuve (Belgium) - jean.vanderdonckt@uclouvain.be

ABSTRACT
This paper presents ERGOSIM, a software that automatically evaluate the design of menu bars, pull-down menus, and sub-menus of a graphical user interface by reviewing usability guidelines related to menu design. In this method, a menu design is parsed against the definition of usability guidelines in order to detect potential usability problems manifested by any occurrence where a guidelines is not respected. Four evaluation strategies are enabled depending on the end user's preferences: an active strategy initiated by the system, a passive strategy initiated by the designer, a mixed strategy collaboratively initiated by both the designer and the system, and a strategy by conceptual units based on the domain. From an initial corpus of 312 usability guidelines compiled from different sources on menu design, a final knowledge base of 58 implemented usability guidelines has been obtained for automatic evaluation. By examining how each usability guideline for menu design is expressed, we discuss to what extent such guidelines could be automated in an automated process by guidelines review.

Author Keywords
Automatic evaluation; computer-aided design; evaluation strategy; heuristic evaluation; menu bar; pull-down menus; sub-menus; usability guidelines.

ACM Classification Keywords
Human-centered computing, Graphical user interfaces. *Human-centered computing - Heuristic evaluations.* Human-centered computing - User interface management systems

INTRODUCTION
In order to assess the usability of a User Interface (UI) and therefore to improve it, the temptation has been followed since years to replace a human (manual) evaluation of this usability by a system (automatic or semi-automatic) evaluation for several reasons [8,13,19,31]: to reduce human resources (e.g., by being released from involving usability experts), to reduce budget resources (e.g., by reducing the time and the resources needed to conduct such an evaluation), to guarantee the quality of the results (e.g., to ensure consistency across several evaluations, even if automated evaluation cannot cover all aspects, to establish a systematic evaluation by reducing missing spots, to minimize false positive and false negative), or to give the label of a usability standard (e.g., by certifying that a particular UI is compliant with a style guide, a corporate design guide or an established standard).

These reasons are considered even more important when the UI has some special status: a UI for a safety-critical system [32] for which it is crucial not to miss any potential defect, a very large UI for which there are so many screens that evaluating them becomes too tedious and repetitive [10], an adaptive UI for which adaptation could give rise to many different configurations to evaluate [11]. Many variables need to be decided when automatically evaluating a UI [8, 14, 19]:

- *What type of method*: several evaluation methods that are good candidates for conducting an automated evaluation, but they do not give all the same type of results. For instance, heuristic inspection, standard compliance, guideline review, cognitive walkthrough, user testing.
- *What type of software*: on-line vs off-line software or mixed solutions exist in order to capture UI data at run-time as well as users or contextual data or not.
- *What type of usability knowledge*: usability guidelines as well as accessibility guidelines are two representative examples of usability knowledge used for guideline review.
- *What type of UI*: as opposed to stand-alone UIs which are more difficult to grasp regarding their code access, web UIs are in principle easier for accessing the HTML code, parsing it, and conducting evaluation. Stand-alone UIs have other sources, like log files, resource files, screen analysis, static analysis, dynamic analysis.
- *What type of scope*: the scope of the evaluation could be also very different, ranging from local evaluation of typed UI elements to global evaluation of all UI elements, including their presentation, navigation, and their contents.

What type of purpose: detecting usability problems or certifying that there should not be such usability problems are two inverse approaches. Between exists the wish to have a simple diagnosis to assess the current usability of a UI in order to locate its quality with respect to competitors.

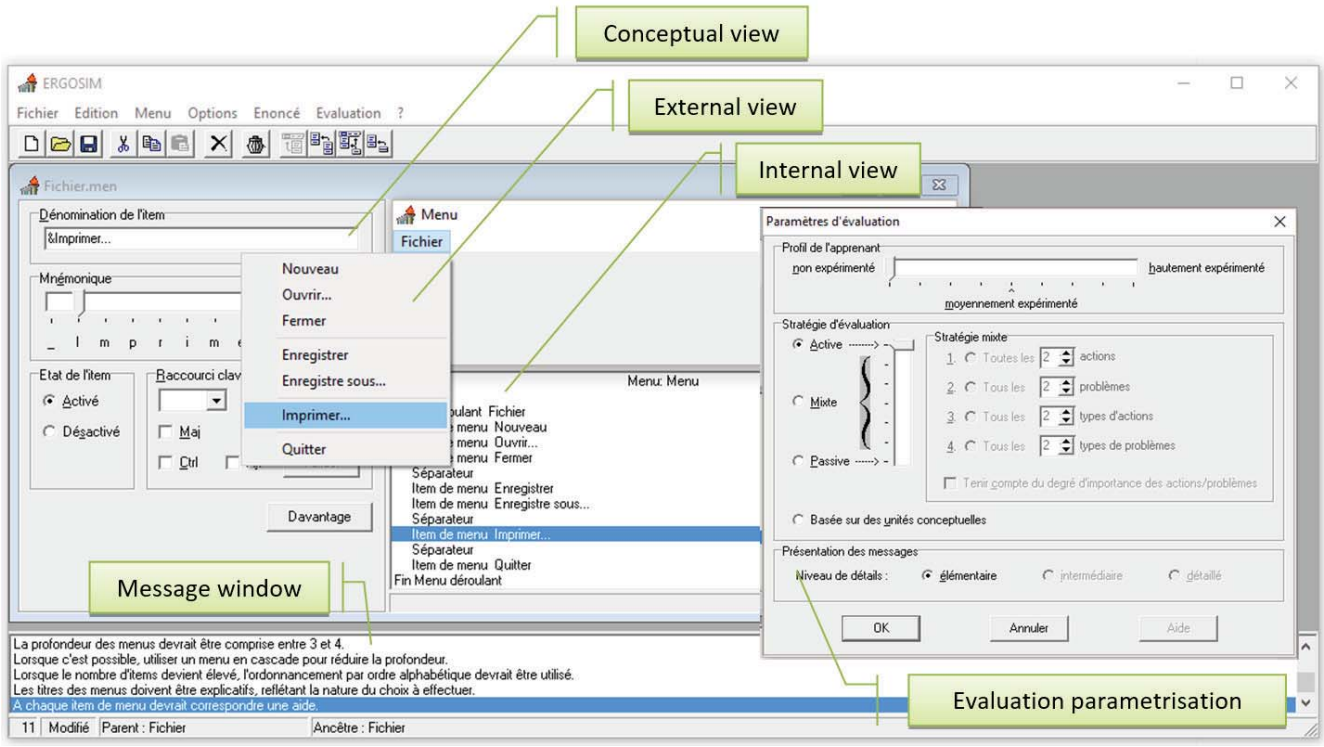

Figure 1. ERGOSIM main screen.

This paper presents ERGOSIM (Figure 1), a software for automated usability evaluation of the menu of a Graphical User Interface (GUI) with the following original aspects:

- *Type of user interface*: a GUI is the focus of the application with standard menus, no adaptable or adaptive menus are considered since they stem for other evaluation methods. Although ERGOSIM is developed on the MS Window platform, it is expected that any IBM Common User Access (CUA)-compliant menu is targeted by ERGOSIM, therefore not assuming that a particular operating system is required to design the menu.
- *Type of method*: guideline review has been decided in order to confront the GUI against a set of usability guidelines that belong to the literature. It is expected that the menu is built at design time (not at run-time) by a designer or a developer. Guideline review [23] consists of selecting a set of relevant guidelines and to examine a UI against this set of guidelines [32]. Two models are prevalent [4]: a binary model where a guideline is considered violated when there is at least instance on the GUI where the guideline is not respect or a linear model where all occurrences of guideline violations count per screen, along with a weight expressing the level of importance of the guideline. The binary model is mainly used here, but with different strategies that will be detailed.
- *Type of software*: a stand-alone application has been decided to enable the designer to build the menu bar, the pull-down menus and the sub-menus during the development phase of detailed design. In order to preserve continuity with the rest of the development life cycle, ERGOSIM can export a menu design as a resource file to be included in a Windows application project. Other export formats, like UsiXML, could be imagined as well, but are not covered. Similarly, it could be imagined that a resource file could be imported for further evaluation.
- *Type of scope*: ERGOSIM focus on only one part of the GUI: the menu bar with its pull-down menus, cascading menus and sub-menus. Several reasons motivate this choice: the menu has never been covered per se by automated evaluation, the menu is probably one of the most frequently used interaction technique in many interactive applications and systems [2], many different types of menu exist [2] although this paper is not aimed at evaluating them all, there is a significant body of knowledge on menu [22], many usability guidelines are widespread in the literature, menu design is a familiar design activity and the menu is an object that could be easily controlled.
- *Type of purpose*: the goal of ERGOSIM is to support the designer while designing the menu, not to conduct an evaluation afterwards when the entire GUI is developed and to help novice designers learning usability knowledge regarding menu design in context [18, 26].

In order to introduce ERGOSIM and to explain how menu usability guidelines are automatically evaluated, the remainder of this paper is structured as follows: Section 2 will review some selected contributions in the area of UI automatic evaluation without conducting a systematic literature review, Section 3 will elaborate on the design and the implementation of ERGOSIM, Section 4 will discuss to what extent usability guidelines have been

implemented in ERGOSIM, and Section 5 will conclude the paper by discussing future avenues to this work.

RELATED WORK

There are many pieces of work related to automatic UI evaluation in general, like Ivory's state-of-the-art [19], although it is no longer up-to-date. A good review is provided in [8, 14]. In this section, we only review some selected work with a focus on menus.

METROWEB [9] enables the designer to access to one or many usability knowledge bases that are presented as hypermedia with faceted search. A typical knowledge base consists of guidelines of any type, along with its ergonomic criteria [5], its linguistic level [30], its impact factor, and positive/negative examples illustrating good and bad practice related to the guideline. Multiple knowledge bases could be accessed and a faceted search could query these bases like "Give me all guidelines related to menu design" by selecting appropriate values for search criteria [30]. Selected guidelines could then be exported in a special section, e.g., for producing an evaluation report. Although METROWEB provides adequate access to usability knowledge, it is the designer's responsibility to correctly apply them or evaluate them. It has been demonstrated that designers relying on METROWEB manipulate more usability guidelines than without and that the UI resulting from this exercise satisfy more guidelines than without [9].

ERGOVAL [17] is a pioneering attempt to automatically evaluate GUIs against usability guidelines (by guideline review) for stand-alone applications. The authors report in their feasibility analysis that a ratio of **40%** has been reached between the guidelines candidates to automated evaluation and their feasible final implementation. They ask the question: what is the limit of automated evaluation?

BOBBY [10] automatically evaluate accessibility guidelines of web sites by guideline review of W3C accessibility guidelines. A pilot study revealed that the ratio could reach up to **50%** for accessibility of web sites since the HTML code of web pages is in principle easily accessible. Nowadays, this ratio is no longer that high with dynamic web pages and CSS3 style sheets [24]: it is around **30%** according to a qualitative estimation.

KWARESMI [4] also automatically evaluate usability and accessibility guidelines of web pages, either on-line or off-line, by guideline review. This process is structured as follows: any candidate guideline is first encoded in GDL (Guideline Definition Language), a XML-compliant language for specifying a guideline based on first-order predicate logic on HTML tags, then incorporated into an evaluation base that is then parsed on-demand for a set of web pages. The advantage is that the evaluation engine is independent of the guidelines encoded in one or many knowledge bases [29]. Although no empirical study has been conducted yet on this software, it is also estimated that a ratio of **30%** of automatable guidelines could be reached, but with different types of restrictions depending on the tags involved in the GDL rule.

Several other software follow the same principle, such as ErgoManager [1], ErgoCoIn [21], MAUVE [29], with a higher degree of flexibility when it supports dynamic web sites instead of a current version of a web page at run-time.

RITA [6] provides a more comprehensive framework for automatic GUI evaluation by considering not only a large set of guidelines, but also by relating them to quantitative data, such as task execution time, task completion rate, error rate, and interaction traces. In this way, RITA establishes a bridge between a qualitative evaluation based on guidelines review and a quantitative evaluation based on metrics.

EISEVAL [15] automatically evaluate usability guidelines on the GUI of an interactive application implemented according to the paradigm of a multi-agent software architecture. In this way, the evaluation consists of a set of autonomous agents which can query different parts of the interactive application so as to gather date and establish a diagnosis based on these data. Multiple agents could be incorporated that conduct different types of evaluation independently of each other, perhaps also with the same guidelines or different ones. In [7], a system is presented that hold the evaluation logic in the very right widgets used by the end user, instead of other modules of the interactive applications.

MENUSELECTOR [25] is a software for rapid prototyping of menu designs by considering different physical parameters like location, orientation, selection mechanism, and group clustering. This approach is purely syntactical since there is no automatic evaluation of the menu being designed, but the automatically generated HTML code could be subject to a further analysis conducted in another software.

MENUDESIGNER [28] is aimed at automatically generating a menu bar, associated cascading menus and menus items based on an activity chaining graph representing possible hierarchical navigation based on a task model. This approach remains static (the menu structure is generated once for all), without any adaptation and could lead to inconsistent menus when items are arranged.

MENUOPTIMIZER [3] is aimed at helping designers and developers to optimize the menu structure by maximizing consistency vs performance based on ant colony algorithm. While MENUOPTIMIZER reveals the popularity of menu items by a color line under each menu item, thus leaving the menu structure untouched, it does not provide end users with an adaptive menu. Matsui & Yamada [20] relied on a genetic algorithm to generate a menu structure that is optimized for its usage.

Adaptivity Animated transitions [12] have also been successfully used to explain to the end user how a UI has been adapted, including for menus [12]: each adaptation operation performed on a GUI is captured, scripted and could be played or replayed at the end user's pace, thus providing some visual explanation of the adaptation. The major drawback was the lack of animation control: not all steps should be animated equally to understand.

Usability Category	Guidelines
Optimizing the user experience	29
Hardware and software	4
The homepage	12
Page layout	9
Navigation	27
Scrolling and paging	3
Headlines, titles and labels	18
Links	21
Text appearance	18
Lists	13
Screen based controls (widgets)	27
Graphics, images and multimedia	17
Writing web content	18
Content organization	8
Search	16
Total Guidelines	240

Table 1. Guidelines implemented in USEFul.

The closest work to ErgoSIM is probably USEFul [13, 14], a complete framework for automatic evaluation of web sites against usability guidelines, also by guideline review. Table 1 summarizes the various categories of guidelines implemented in USEFul. Each guideline cannot be implemented with the same level of support. For this purpose, USEful distinguishes three levels of implementation [14]:

1. *Green,* when the guideline can be fully implemented: the framework is able to automatically determine whether this guideline applies to the web site being evaluated and the results related to this guideline are conclusive since these types of guidelines are typically measurable, with clearly defined parameters.
2. *Amber*, when the guideline is harder to fully implement in the USEFul framework: certain patterns have been used in order to determine whether this guideline may apply to the web site being evaluated and then transformed into a corresponding code. This guideline could be upgraded by augmenting the guideline evaluation by other mechanisms than guideline review, such as with machine learning processes or artificial intelligence algorithms. The results provided by USEFul for this guideline consist of data that can assist the designer in checking whether it applies to the web site being evaluated or not and support the designer in conducting the evaluation of this guideline which required human interpretation.
3. *Red*, when the guideline is too abstract to warrant any implementation and requires user intervention or too advanced algorithms to make it possible for it to be implemented in the framework. Through the use of such sophisticated algorithms, a guideline could be upgraded to “amber” or “green” levels. In its current definition, USEFul lists this guideline so that the designer can be manually checked if it applies to the web site being evaluated.

In conclusion, one can observe that today there is no software like ERGOSIM to perform automatic evaluation of usability guidelines related to menus of GUIs of interactive applications at design time. Some software could be however tailored for this purpose, although they mainly work over web applications for which the HTML code is downloadable as opposed to a stand-alone application.

DEVELOPMENT OF ERGOSIM

Design options

This section presents the major design options decided for developing ERGOSIM and discusses the rationale behind.

Multi-view visualization. We hereby define a *UI view* as any representation of a final UI involved in a development life cycle. A UI view may be textual, graphical or both, based on a data structure or not [12]. By observing existing UI development methods and development life cycles, UI views can be roughly classified into three categories (Figure 2):

1. *Conceptual View* (CV): describes a conceptual representation of a UI of interest based on semantics, syntax, and stylistics. Typical examples include: UI models for domain, functional core, resources, and dynamic aspects. A conceptual view is the designer’s view at early stage.
2. *Internal View* (IV): consists of the UI code in any programming or markup language. An internal view is the typical developer view for developing a particular UI.
3. *External view* (EV): refers to the final UI that is visible and executable by the end user.

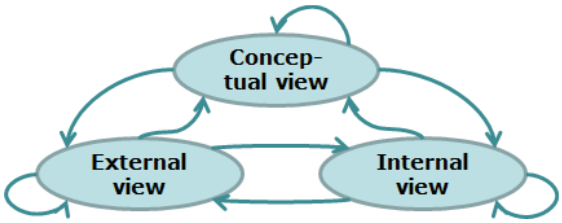

Figure 2. Possible paths between UI views.

During the development life cycle, at design-time as well as at run-time, various UI stakeholders can create, retrieve, modify, delete, or simply execute any UI view or view element: for instance, while a designer is responsible for the conceptual view, the developer is responsible for the internal view, and the end user accesses the external view for comments, testing, and validation. A development path may be initiated from any view and could proceed with any other view, including itself, which are respectively represented by arrows and loops in Figure 2. ERGOSIM structures its environment similarly into three views (Figure 1) [12]:

1. *External view* (EV): refers to the final representation of the menu as it is visible and executable by the end

user with the same Look & Feel as it should be in the end.

2. *Internal View* (IV): consists of the menu structure decomposed into levels of the menu hierarchy, as it is stored for instance in a resource file.
3. *Conceptual View* (CV): describes a conceptual representation of a menu in terms of design options and parameters for each menu item or menu group, which includes:
   a. The *menu label*, which contains the textual label of the menu item, along with the "&" character representing the mnemonic of the menu item. The character after this delimiter is underlined.
   b. The *mnemonic* of the menu item, which is the character to be pressed by combining it with the "Alt" key instead of selecting it by pointing, e.g. "Alt + 3 for "Save". The range of possible mnemonics for a label is automatically generated from the menu label (Figure 3) and the designer can choose among them by moving a cursor on it. This is important since a usability guideline states that a mnemonic should always be chosen among the real letters of the label, preferably those that are pronounced.
   c. The *menu activation status*, qui specifies whether the item is by default activated or deactivated (greyed).
   d. The *menu shortcut*, qui defines the sequence of keys to be pressed for directly accessing the menu item, which consists of normal keys, i.e., A, B, C, ..., X, Y, Z, 1, 2, ...9, 0, F1, F2, ..., F11, F12, Del, Ins, ... and control keys, i.e., « Crtl », « Alt », « Shift ».
   e. The *menu attachment type*, which specifies whether a menu item is related to displaying a sub-menu (for instance, a pull-down menu or a cascading menu), to opening a dialog box or a secondary window, or to triggering directly a function of the application.
   f. The *contextual help message*, which specifies the message to be displayed in the status bar when the menu item is highlighted, but not yet selected.

The menu bar is consequently equipped with traditional facilities for menu management, such as creating, updating, deleting a menu item, a group of items, an entire menu or a menu bar. Note that this conceptual view could be expanded in the future with other parameters, such as the associated earcon or the gesture to trigger the same item, but these options are not covered yet by usability guidelines.

Multi-strategy evaluation. The target users of ERGOSIM, theoretically any stakeholder involved in the UI development life cycle but practically the most often, designers and developers, could exhibit very different profiles in terms of background and level of experience [26].

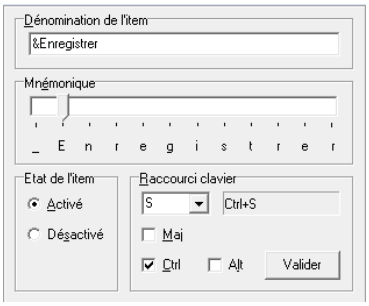

Figure 3. Conceptual view of a menu item.

In order to support this variation, the automatic evaluation of the menu being designed could be achieved flexibly according to parameters specified in the *Evaluation parametrization* window (right part of Figure 1), which offers four evaluation strategies:

1. An *active evaluation strategy*, where the end user decides when and how the automatic evaluation will take place. This strategy is qualified as "active" because the stakeholder actively participates in the evaluation.
2. A *passive evaluation strategy*, where the system automatically evaluates the menu being designed without any intervention of the end user. This strategy is qualified as passive since the stakeholder has no control on the evaluation process.
3. A *mixed initiative strategy*, which is located mid-way between the active and the passive strategies, where the end user can parametrize the evaluation based on several parameters:
   a. The *amount of user actions*: which captures the amount of all elementary actions performed by the user, such as menu item editing, pull-down menu editing, etc. In this way, it is possible to trigger the evaluation every 5 actions.
   b. The *type of user actions*: which categorizes the level of actions performed by the end user: elementary, intermediate, or complete. In this way, it is possible to trigger the evaluation when a complete pull-down menu is finished, therefore not interrupting the user in the design process. Task switching between a design activity and an evaluation activity should be minimized.
   c. The *amount of usability problems*, which specifies the amount of usability problems detected after which the evaluation could be triggered. In this way, it is possible to trigger an evaluation after a certain amount of problems has been detected, which is particularly useful when problems are generated in cascade: one usability problem may immediately induce some other related problems.
   d. The *type of usability problem*, which specifies the level of importance of a detected violation of a usability problem, ranging from 1 (cosmetic) to 5 (critical). The level of importance of a guideline is stored in its definition or automatically suggested from the linguistic level: the higher the linguistic level is, the higher becomes the level of

importance. In this way, it is possible to trigger an evaluation when a problem with a given severity is detected, and not just after any occurrence of a detected violation.

4. A *strategy based on conceptual units*, where the end user decides when the system should evaluate significant parts of the menu being designed, based on conceptual units. A *conceptual unit* is defined as a non-elementary menu group unit, such as an entire group of menu items delineated by separators in a pull-down menu, an entire pull-down menu, a cascading menu or the whole menu bar. In this way, the evaluations could be triggered as soon as a significant part of the menu has been completed, not before. This strategy is qualified as "based on conceptual units" since the evaluation scope is on a menu part that has some semantic meaning, not an elementary item.

Note that for the moment in the mixed-initiative strategy, the evaluation is triggered only based on simple conditions with a threshold, such as when the amount of problems >= 5 or when 2 important problems have been detected. A cursor between these strategies (Figure 1) enables the end user to gracefully evolve between strategies.

Parametrizable feedback. In addition to the evaluation strategy, the end user may want to specify the *level of feedback detail* that governs the way feedback messages are presented to the end user after an evaluation has been performed. This level of feedback could be stated to [16]:

- *Elementary*, when only the short title of the usability guideline violated is presented for each occurrence of a usability problem, along with its location (see the message window at the bottom of Figure 1).
- *Intermediate*, when the complete title of the guideline violated is presented for each occurrence of a usability problem, along with its location and the level of importance.
- *Detailed*, when the message contains the full set of information on any detected usability problem: the complete title, the ergonomic criteria from Bastien & Scapin [], the linguistic level, positive and/or negative examples, references where the guideline is documented, along with information on the location and a possible help on how to fix it.

Any occurrence of a detected usability problem is displayed in the message window with or without a timestamp. The message window could be purged at any time via an appropriate push button. The amount of information displayed in the message window can be tailored (Figure 4).

Management of user profiles. A user profile could be created and updated at any time that captures the parameters:

The *level of experience*, which specifies the level of usability experience in general and more specifically for menu design: low experience, medium experience, or high experience. Based on this value, ERGOSIM can automatically assign predefined values to other parameters, like the evaluation strategy so that the end user should not necessarily fill in all the parameters before starting. In this way, if the end user selected "low experience", ERGOSIM will pick the passive evaluation strategy and the elementary feedback. It the end user estimates herself as "moderately experienced", ERGOSIM will pick the mixed-initiative evaluation strategy with a feedback every 5 significant actions. If the end user estimates herself "highly experienced", ERGOSIM will pick the active strategy. The user can change the values of these parameters at any time.

- The *evaluation strategy* that is preferred by the end user, according to the aforementioned definition.
- The *level of feedback detail* that is preferred by the end user, according to the aforementioned definition.
- The evaluation parametrization options (Figure 5).

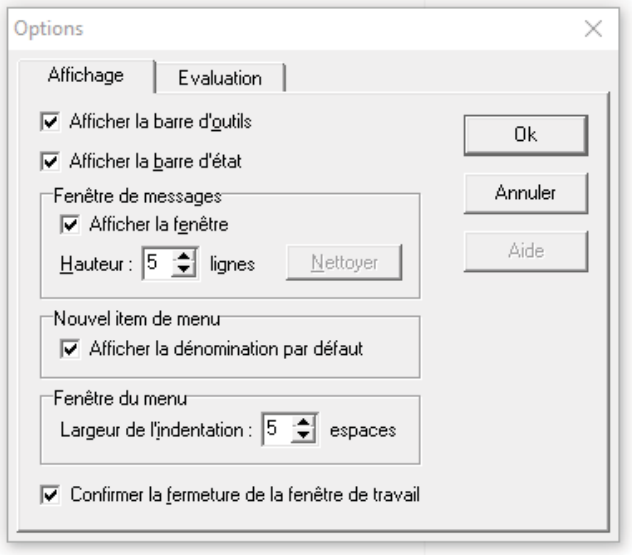

Figure 4. Evaluation display options.

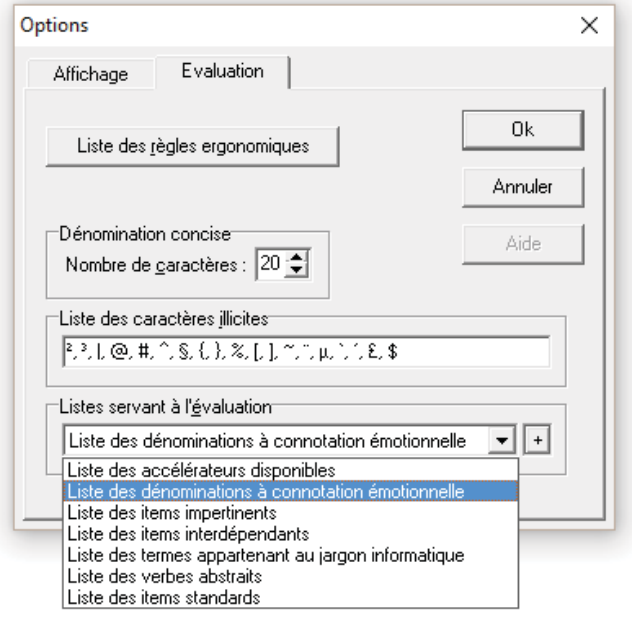

Figure 5. Evaluation parametrization options.

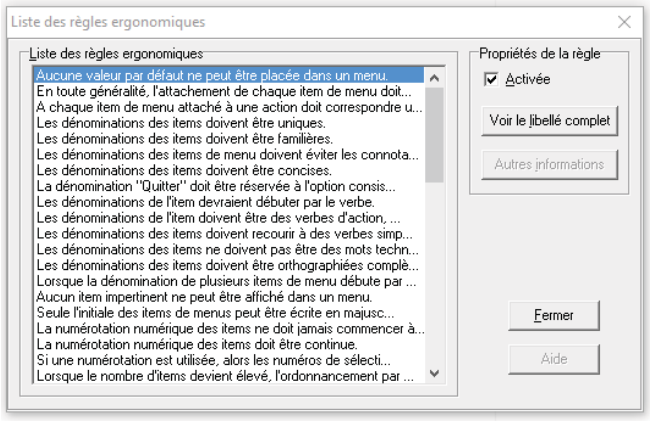

Figure 6. Activation of selected guidelines.

Evaluation parametrization options (Figure 5) enable the end user to tailor various options that drive the evaluation of guidelines themselves:

- The *list of possible guidelines*: any guideline can be activated or de-activated momentarily and this configuration can be saved in a configuration file (Figure 6).
- The *list of inappropriate terms*: since no natural language understanding is incorporated, the end user may want to specify a series of terms that hold a negative connotation, whose usage is therefore prohibited. For instance, "Abort" in English is inappropriately translated into "avorter" in French, which is irrelevant (Figure 7).
- The *list of interdependent terms*: for the sake of the evaluation based on conceptual units, a series of constraints could be imposed to establish and maintain semantic relationships between terms that have some interdependency. For instance, "Save" and "Save as" should be located one after another, "Open" and "Close" or synonyms should be grouped in a same group of menu items to convey disclosure.
- The *list of computer-based terms*: this includes terms that are considered as jargon terms belonging to the area of computer science, such as "bandwidth", "baud rate", "bitmap", "memory dump". These terms should be avoided.
- The *list of abstract terms*: in order to evaluate the general guideline stating that a menu item should be ideally structured in a simple sentence composed of an action verb followed by an object on which the action is executed (action-object paradigm) or vice-versa (object-action paradigm), this list contains verbs that are considered too abstract or generic to be used in appropriate menu design.
- The *definition of standard menu items*: standardized menus as found in standards like IBM Common User Access (CUA), in software vendors or operating systems style guides (e.g., MacOs, Ubuntu, MS Windows) can be defined once for all in a profile so as to be compliant with these sources (Figure 7).

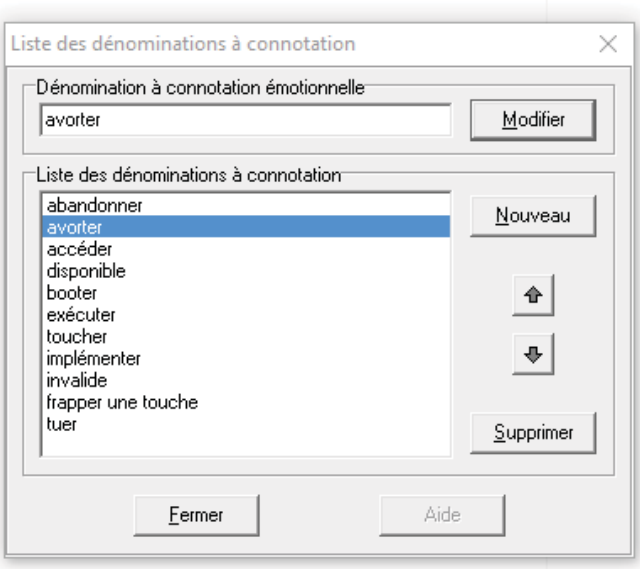

Figure 7. Definition of inappropriate labels.

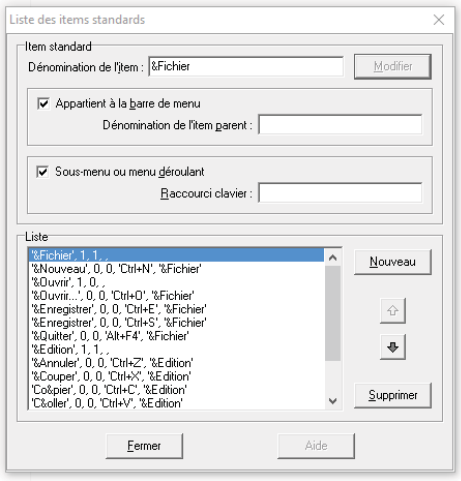

Figure 8. Definition of standard items.

Development of ErgoSim

ERGOSIM has been developed in Borland Pascal for Windows 7.0 because of the object-oriented facilities offered by the procedural language and its corresponding environment, but also for the object-oriented database in which each usability guideline will be stored as a record. The software architecture of ERGOSIM is composed of three modules:

1. The *evaluation triggerer*: this module receives as input any action executed by the end user performed on the internal or the conceptual view of the menu and the values of options contained in the user profile, the most important being the evaluation strategy with its parameters. This module then triggers an evaluation of the menu being designed based on the evaluation strategy and other parameters on the end users actions performed since the last evaluation. These actions include, but are not limited to: modifying the label of a menu item, inserting a pull-down menu, inserting a new menu item, modifying the menu bar, defining the shortcut of a menu item, moving a group of items from one sub-menu to another menu, moving a group of items to a sub-menu, using standard menu items in their standard format.

2. The *evaluation engine*: this module receives as input a knowledge base of usability guidelines and the internal representation of the menu from the triggerer and performs the evaluation according to parameters set by the triggered to return the results of the evaluation. After an evaluation has been performed, all parameters regarding the amount of actions, problems, etc. is reinitialized. The system does not keep trace of usability problems that are not solved: it simply re-checks them at any evaluation. The evaluation engine is independent of the knowledge base containing the evaluation logic.
3. The *evaluation presenter*: this module receives from the evaluation engine the results of a performed evaluation and produces the output according to the evaluation display options (Figure 4) and user profile. The results are displayed in the message window (bottom right of Figure 1).

AUTOMATIC EVALUATION OF USABILITY GUIDELINES

Although ERGOSIM could accommodate one or several different knowledge bases, it was decided to compose one comprehensive knowledge base containing all the possible guidelines on menu design. For this purpose, we compiled usability guidelines from two major sources: Scapin's *guide ergonomique* [23] and Vanderdonckt's *ergonomic guide* [27], which is itself a compilation of usability guidelines coming from more than 300 sources delivering usability guidelines. This compilation resulted into a base of 362 unique usability guidelines (without double entries), which is considered as the set of initial guidelines subject to automatic evaluation.

From this initial set, only 58 usability guidelines out of 362, have been finally implemented, which represents a ratio of **16%**. If we count usability guidelines that are intrinsically respected by the operating system, the software environment of ERGOSIM or ERGOSIM itself due to its implementation (e.g., some guidelines are intrinsically respected when displayed in the external view), this ratio reaches to **36.5%**.

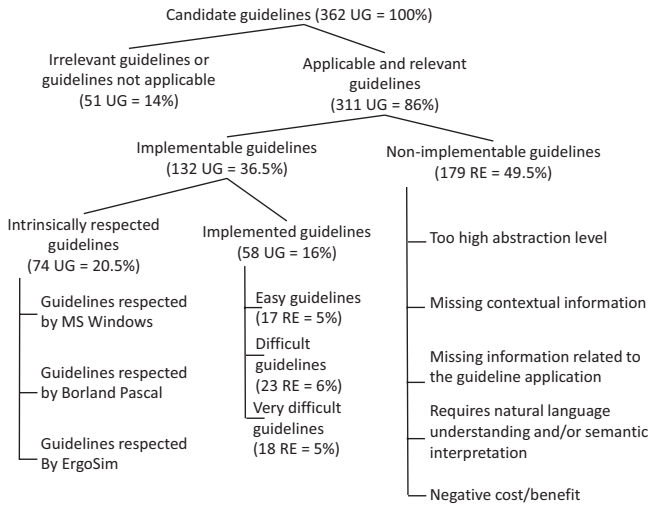

Figure 9. Distribution of implemented usability guidelines.

The full distribution of usability guidelines is graphically depicted in Figure 9. Usability guidelines fall into 4 categories depending the level with which they could be implemented, similarly to the USEFul's level of implementation [14]:

1. *Irrelevant or not applicable guidelines*: this category contains guidelines whose application is probably relevant to menu design in general, but not for menu bar, pull-down menus, and cascading menus or guidelines that cannot be applied practically. Table 2 reveals some significant examples of such guidelines along with a comment explaining why they cannot be applied.
2. *Non-implementable guidelines*: this category contains guidelines whose interpretation and/or application is impossible to replicate by a software for different reasons (Figure 9): guidelines are expressed at a too high level of abstraction that prevent them to be interpreted by an automaton, guidelines that require additional information related to the user, the platform, the environment or the whole context of use that is unknown at design time, guidelines that require additional information that cannot be obtained by any means, guidelines that require understanding of the natural language in which the guideline is expressed, and guidelines whose development would be so complicated that they would require a significant amount of time for a small benefit. Note that these reasons are independent of the environment in which ERGOSIM is implemented. Table 3 reveals some significant examples of such guidelines.
3. *Intrinsically respected guidelines*: this category contains guidelines that are intrinsically respected either by the operating system (here, MS Windows), the development platform (here, Borland Pascal) or the environment of ERGOSIM itself. For instance, the guideline "every menu item should be either activated or deactivated" is automatically ensured by the conceptual view of ERGOSIM. Similarly, the guideline "Shortcuts should always be visible" is straightforwardly ensured by MS Windows. The guideline "A main menu should always exist" is also intrinsically established by MS Windows since a menu bar is always created, even if minimal. The guideline "Shortcuts should always made visible" is ensured by ERGOSIM itself since the conceptual view automatically propagates this design choice on the external view, thus making them visible automatically. The guideline "Menu items should be perceptually distinct from each other" is ensure by both MS Windows and Borland Delphi since menu items in the external view always presented with the same space between and separators defined by the end user in the group.
4. *Implemented guidelines*: this category contains guidelines implemented in some way in ERGOSIM, which are further refined into three sub-categories depending on their complexity and the level with which the scope of the guidelines could have been addressed (Figure 9):

a. *Easy guidelines*: guidelines that are straightforwardly implemented. For instance, "The menu breadth should not exceed 8 items" required 17 Lines of Code (LOC) in Borland Pascal, "The menu items should have unique labels" required 28 LOC.
b. *Difficult guidelines*: guidelines that are implemented but with some restrictions in their interpretation [30]. For instance, "The numbering of menu items should continuous" required 60 LOC, "Menu mnemonics should be phonetically distinct" required 43 LOC, based on an existing SOUNDEX algorithm testing whether two strings are phonetically close or not.
c. *Very difficult guidelines*: guidelines that are implemented with advanced techniques or significant restrictions. For instance, "Mutually exclusive items or interdependent items should be grouped together" required 76 LOC, "Menu items should avoid abstract terms and prefer action verbs" is only 26 LOC because it merely tests that all items do not belong to a list of predefined terms considered abstract or not.

Guideline statement	Reason
Menu items should be consistent from one application to another	ERGOSIM evaluates one menu design at a time and cannot compare with other menu design for other case studies.
Full screen menus should be displayed at once, with one item per line	ERGOSIM does not cover full screen menus
Linear menus should match user's expectations	There are no linear menus in ERGOSIM
Network menus should follow a natural flow	There are no network menus in ERGOSIM
Contextual menus should be displayed at their right location (top, bottom, left, right) depending on the task	ERGOSIM is focusing on menu bars, pull-down menus and cannot relocate such menus at different locations
Items of pull-down menus attached to a label of the menu bar could be colored in the same way	This guideline is mostly applicable to web sites and item coloring is an unsupported feature

Table 2. Examples of irrelevant guidelines.

Guideline statement	Reason
The menu design should be based on a metaphor of a mini-world, based on real task options	Impossible to interpret unless a model of the mini-world is available along with a task model
Use menu selection technique that is precise enough	This implies to rely on a menu performance model, which exists, but is another input.
The complexity of menus should reflect the end user's experience	This requires to access a user model
Only action verbs of natural language should be used	This requires a thesaurus of action verbs for the natural language used
Menu items should have unique meanings	This requires a module for natural language processing based on a semantic network
Menu items should avoid any humor	This requires an interpreter of natural language

Table 3. Examples of non-implementable guidelines.

CONCLUSION

This paper presents ERGOSIM, a software that automatically evaluate the design of menu bars, pull-down menus, and sub-menus of a graphical user interface by reviewing usability guidelines related to menu design in a design-time environment, thus preserving the continuity between design activities and evaluation activities. ERGOSIM automatically evaluate 16% of usability guidelines for menu, or 36% if we count intrinsically respected guidelines. This is rather different from ERGOVAL [17], whose authors argue that the ratio should be between 44% and 78% or from BOBBY [10], whose authors argue that a ratio of 50% was reached. In the last case, web sites were automatically evaluated against usability and accessibility guidelines, which is considered as an easier case since the HTML code is accessible, perhaps also with the Document Object Model (DOM) containing the structure of the web page and the CSS. A closer observation of guidelines that are finally supported by the whole environment could be classified as follows by linguistic level:

- Guidelines belonging to the physical and alphabetical levels are almost always established by construction of the menu bar, the pull-down menus and the cascading menus. Changing the alphabet is also possible, but in another environment.
- Guidelines belonging to the lexical level are almost all supported since they are all easy to implement.
- Guidelines belonging to the syntactical level are often supported, sometime with a more advanced technique.
- Guidelines belonging to the semantic level could be sometimes implemented provided that some restriction, e.g. by replacing the full scope by a list of admissible values, is adopted.
- Guidelines belonging to the pragmatic and the goal levels are almost never possible to implement, unless additional models are made accessible, thus required artificial intelligence techniques, such as intelligent model-checking techniques, machine learning techniques, relevance feedback or reinforcement learning.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their constructive feedback on an earlier version of this paper and Grenoble INP for supporting this collaboration.

REFERENCES

1. de Abreu Cybis, W., Scapin, D.L., Morandini, M. ErgoManager: A UIMS for monitoring and revising user interfaces for Web sites. In *Proc. of the 1st Int. Conf. on Web Information Systems and Technologies* (WEBIST'2005, Miami, 26-28 May 2005). SciTe Press, (2005), 281–286.
2. Bailly, G., Lecolinet, E., Nigay, L. MenUA: A Design Space of Menu Techniques (2009). Retrieved from http://www.gillesbailly.fr/menua/ on May 27th, 2016.
3. Bailly, G., Oulasvirta, A., Kötzing, T., and Hoppe, S. MenuOptimizer: interactive optimization of menu systems. In *Proc. of the 26th ACM Symposium on User interface software and technology* (UIST'2013). ACM Press, New York, (2013), 331–342.
4. Beirekdar, A., Vanderdonckt, J., Noirhomme-Fraiture, M. A Framework and a Language for Usability Automatic Evaluation of Web Sites by Static Analysis of HTML Source Code. In *Proc. of 4th Int. Conf. on Computer-Aided Design of User Interfaces* (CADUI'2002, Valenciennes, 15-17 May 2002). Kluwer Academics Pub., Dordrecht, (2002), 337–348.
5. Bastien, Ch. And Scapin, D.L. Evaluating a User Interface with Ergonomic Criteria, *International Journal of Human-Computer Interaction 7*, (1995), 105–121.
6. Charfi, S., Ezzedine, H., and Kolski, Ch. RITA: a useR Interface evaluaTion frAmework. *Journal of Universal Computer Science 21*, 4, (2015), 526–560.
7. Charfi, S., Trabelsi, A., Ezzedine, H., and Kolski, Ch. Widgets Dedicated to User Interface Evaluation. *Int. J. Hum. Comput. Interaction 30*, 5, (2014), 408–421.
8. Charfi, S. and Ezzedine, H. Evaluation tools through user participation techniques: Features, limitations, and new perspectives. In *Proc. of Int. Conf. on Advanced Logistics and Transport* (ICALT'2014, Hammamet, 1-3 May 2014). IEEE Press, (2014), 13–18.
9. Chevalier, A., Fouquereau, N., and Vanderdonckt, J. The influence of a knowledge-based system on designers' cognitive activities: a study involving professional web designers. *Behaviour and Information Technology 28*, 1, (2009), 45–62.
10. Cooper, M., Limbourg, Q., Mariage, C., and Vanderdonckt, J. Integrating Universal Design into a Global Approach for Managing Very Large Web Sites. In *Proc. of the 5th ERCIM Workshop on User Interfaces for All* (UI4All'99, Dagstuhl, 28 November-1 December 1999). A. Kobsa, C. Stephanidis (Eds.). GMD Report 74, GMD - Forschungszentrum Informationstechnik GmbH, Sankt Augustin, (1999), 131–150.
11. Dessart, C.-E., Genaro Motti, V., and Vanderdonckt, J. Showing user interface adaptivity by animated transitions. In *Proc. of ACM Conf. on Engineering Interactive Computing Systems* (EICS'2011). ACM Press, New York, (2011), 95–104.
12. Dessart, C.-E., Genaro Motti, V., and Vanderdonckt, J. Animated transitions between user interface views. In *Proc. of ACM Working Conf. on Advanced Visual Interfaces* (AVI'2012). ACM Press, (2012), 341–348.
13. Dingli, A. and Cassar, S. An Intelligent Framework for Website Usability, 2014. *Advances in Human-Computer Interaction 2014*. DOI : http://dx.doi.org/10.1155/ 2014/479286
14. Dingli, A. and Mifsud, J. USEFul: A Framework to Mainstream Web Site Usability through Automated Evaluation. *International Journal of Human Computer Interaction 2*, 1, (2011), 10–30.
15. Dung Tran, Ch., Ezzedine, H., and Kolski, Ch. EISEval, a generic reconfigurable environment for evaluating agent-based interactive systems. *Int. Journal Human-Computer Studies 71*, 6, (2013), 725–761.
16. Ericsson, M., Baurén, M., Löwgren, J., and Y. Wærn. A study of commenting agents as design support. In *Proc. of ACM SIGCHI Conference on Human Factors in Computing Systems* (CHI'1998). ACM Press, New York, (1998), 225–226.
17. Farenc, Ch., Liberati, V., Barthet, M.-F. Automatic Ergonomic Evaluation: What are the limits? In *Proceedings of 2nd International Workshop on Computer-Aided Design of User Interface*s (CADUI'96). Presses Universitaires de Namur, Namur, (1996), 159–170.
18. Fischer, G., Lemke, A.C., Mastaglio, T.W., and Mørch, A.I. The Role of Critiquing in Cooperative Problem Solving. *ACM Trans. Inf. Syst. 9*, 2, (1991), 123–151.
19. Ivory, M.Y. and Hearst, M.A. The state of the art in automating usability evaluation of user interface. *ACM Computing Surveys*, (2001), 470-516.
20. Matsui, S. and Yamada, S. Genetic algorithm can optimize hierarchical menus. In *Proc. of the ACM SIGCHI Conference on Human Factors in Computing Systems* (CHI'2008). ACM Press, NY, (2008), 1385–1388.
21. Morandini, M., de Abreu Cybis, W., and Scapin, D.L. A Prototype to Validate ErgoCoIn: A Web Site Ergonomic Inspection Technique. In *Proc. of HCI International'2009*, (2009), 339–348.
22. Norman, K.L. *The Psychology of Menu Selection*, Designing Cognitive Control at the Human/Computer Interface. Ablex Publishing Corporation, Norwood, New Jersey, 1991.
23. Scapin, D.L. *Guide ergonomique de conception des interfaces homme-ordinateur*. Research report INRIA n°77. Institut National de Recherche en Informatique et en Automatique, Le Chesnay, 1986.
24. Schiavone, A.G. and Paternò, F. An extensible environment for guideline-based accessibility evaluation of dynamic Web applications. *Universal*

Access in the Information Society 14, 1, (2015), 111–132.

25. Spoidenne, J., Vanderdonckt, J. MenuSelector: Automated Generation of Dynamic Menus with Guidelines Support. In *Proc. of 10th Int. Conf. on Human-Computer Interaction* (HCI International'2003). Vol. 1, J. Jacko, C. Stephanidis (Eds.). Lawrence Erlbaum Associates, Mahwah, (2003), 233–237.
26. Sumner, T., Bonnardel, N., and Kallak, B.H. The Cognitive Ergonomics of Knowledge-Based Design Support Systems. In *Proc. of ACM SIGCHI Conf. on Human Factors in Computing Systems* (CHI'1997). ACM Press, New York, (1987), 83–90.
27. Vanderdonckt, J. *Guide ergonomique des interfaces homme-machine*. Presses Universitaires de Namur, Namur, 1994.
28. Vanderdonckt, J. Computer-Aided Design of Menu Bar and Pull-Down Menus for Business Oriented Applications. In *Proc. of 6th Int. Workshop on Design, Specification, Verification of Interactive Systems* (DSV-IS'99). Springer, Vienna, (1999), 73–88.
29. Vanderdonckt, J. and Beirekdar, A. Automated Web Evaluation by Guideline Review. *Journal of Web Engineering 4*, 2, (2005), 102–117.
30. Vanderdonckt, J. Development Milestones towards a Tool for Working with Guidelines. *Interacting with Computers 12*, 2, (1999), 81–118.
31. Xiong, J., Diouf, M., Farenc, Ch., and Winckler, M. Automating Guidelines Inspection: From Web site Specification to Deployment. In *Proc. of CADUI'2006*, 273–286.
32. Xu, J., Ding, X., Huang, K., and Chen, G. A Pilot Study of an Inspection Framework for Automated Usability Guideline Reviews of Mobile Health Applications. In *Proceedings of Wireless Health 2014 on National Institutes of Health* (WH'2014). ACM Press, New York, (2014), 1–8.