Published in: International Journal of Human-Computer Interaction, 30 (5), pp. 408-421, 2014.

Widgets dedicated to user interface evaluation

Selem Charfi
UVHC, LAMIH – UMR CNRS 8201, F-59313 Valenciennes, France
Selem.Charfi@univ-valenciennes.fr

Abdelwaheb Trabelsi
LOGIC, ISIG, 3018, University of Sfax, Sfax, Tunisia
Abdelwaheb.Trabelsi@fss.rnu.tn

Houcine Ezzedine
UVHC, LAMIH – UMR CNRS 8201, F-59313 Valenciennes, France
Houcine.Ezzedine@univ-valenciennes.fr

Christophe Kolski
UVHC, LAMIH – UMR CNRS 8201, F-59313 Valenciennes, France
Christophe.Kolski@univ-valenciennes.fr

Abstract: In this paper, we propose evaluation based widgets as a contribution to assist evaluators for early evaluation of user interfaces. This contribution imbricates the ergonomic quality evaluation process into widgets used for user-interface graphical composition. In other words, these widgets evaluate themselves according to a defined set of ergonomic guidelines. The proposed widgets indicate the possible interface design ergonomic inconsistencies as a notification to the designer. The guidelines set can be modified through an interface dedicated to guidelines definition into XML files. The proposed widgets are intended for the evaluation of different kind of user interfaces: WIMP, Web and Mobile. An experimental evaluation, involving these evaluation based widgets is proposed to illustrate and to validate the approach.

Keywords: Widget, Human-Computer Interaction (HCI), Graphical User Interface (GUI), Ergonomic Guidelines (EG), Interactive System Evaluation

1 Introduction

Software verification and validation is a common practise in the field of software engineering (Sommerville, 2010). Among the tested aspects is the software user interface (UI). The user interface evaluation domain is a very rich domain in terms of research, concepts, tools and techniques[1]. This domain dates back more than forty years (Nielsen, 1994; Wright, Blythe, McCarthy, Gilroy, & Harrison, 2006; Rogers, Sharp, & Preece, 2011; Bardzell, 2011). It consists on improoving and optimizing interactive systems to reduce erroneous, incorrect, inappropriate and ineffective user actions. It is generally based on utility and usability as quality criteria (Nielsen, 1994; Grudin, 1992; Rafla, Robillard, & Desmarais, 2006; Juristo, Moreno, & Sanchez-Segura, 2007; Folmer & Bosch, 2004). In some cases, user interface evaluation is essential such as in the case of critical systems

[1] In several bibliographical resources, authors use the term "evaluation method". In this article we opt for the term "evaluation technique". This choice is due to the fact that a method is generally defined as an ordered set of principals, rules, steps, etc. The technique is defined as a set of processes and practical means for an activity. Thus, we think that "technique" is the term the most adequate due to the fact that generally there is not a well ordered process for user interface evaluation. Typically, evaluation tools are meant to automatically support some underlying evaluation techniques.

(power production, transportation, aeronautics, health care domains, and so on) (Kortum, 2009; Boy, 2011).

In the international Human-Computer Interaction (HCI) community, this research is abundant and revolves essentially around approaches, tools and techniques. Each of them possesses its specificities and requirements. They differ according to many features and mainly according to the application stage of the software development process phase (e.g., waterfall systems development life-cycle phases: requirements, design, implementation, verification and maintenance (Medvidovic & Jakobac, 2005)).

We distinguish essentially four categories of evaluation tools:

- Tools that are used on the interactive systems once finished (e.g., Access Enable by Brinck, Hermann, Minnebo, and Hakim (2002) and EISEval by Tran, Ezzedine, and Kolski (2008));
- Tools that are used by evaluation experts to evaluate advanced prototypes (e.g., Cognitive Walkthrough method and its numerous extensions and variants (Wharton, Bradford, Jeffries, & Franzke, 1992; Mahatody, Sagar, & Kolski, 2010));
- Tools for interface generation that consider diverse usability aspects (Savidis & Stephanidis, 2006; Folmer & Bosch, 2004).
- Tools for evaluating interactive systems since the first development phases (e.g., THEA[2] by Pocock, Harrison, Wright, & Johnson (2001).

The last category of evaluation tools explicitly couples the design phase and the evaluation phase (Nielsen, 1994; Tarby, Ezzedine, & Kolski, 2008). One of its advantages is that the evaluation process is less costly. We do not need to improve and correct the User Interface that has already been implemented. Correcting an already implemented interface can turn out to be expensive in terms of effort and time. According to Nielsen (1994), it can be 100 times more expensive to correct an already designed system than to correct it at the early stages of the systems development life-cycle.

Dix, Finlay, Abowd, and Beale (2003) distinguish mainly two categories of UI evaluation techniques:

- *Evaluation through expert analysis techniques.* These techniques concentrate mainly on evaluating the system design by the designer and/or the expert evaluation. It aims at identifying any aspects than can lead to use difficulties or can violate known cognitive principles. Its main advantage is the fact that the used evaluation process is not costly due to the fact that it does not require testing the system with users. Illustrative examples of such techniques are: Cognitive Walkthrough, Heuristic Evaluation, etc.
- *Evaluation through User participation Techniques.* This techniques set includes empirical techniques, experimental techniques, observational techniques, query techniques, techniques using physiological monitoring. It needs user participation to test the system. The system can be a prototype, at early version or in the final state.

Filippi and Barattin distinguish another categorie that concerns an “hybrid“ category. The associated evaluation techniques involve user and expert during the evaluation process (Filippi & Barattin, 2012).

Although there are many tools for user interface evaluation, evaluators still find difficulties to evaluate UI. First, the evaluation process is complex and difficult to establish in order to identify the UI utility and usability problems (Hearst, 2009). In addition to that, the early evaluation tools are rare. Note that early evaluation tools are mainly structured into three categories : heuristic evaluation, usability principle application and usability tests on system prototype (Hvannberg, Law, & Lárusdóttir, 2007).

[2] It is a technique for designing interactive systems that are resilient to user erroneous actions, in which the evaluation takes place in the first stages of the software development cycle.

Usually to proceed to an early evaluation, evaluators have to conduct the prototyping technique (Leonidis, Antona, & Stephanidis, 2012; Buxton, 2007). Indeed, the prototype implementation is fast and therefore inexpensive. These prototypes are improved and modified until user interfaces conform to specific usability standards (Konstan, 2011). Early evaluation requires one or more experienced evaluators to exploit ergonomic guidelines (or heuristics) for UI evaluation (Salvendy & Turley, 2002). Among the early evaluation existing approaches, we can mention Tarby early evaluation approach (Tarby et al., 2008). It is based on aspect oriented programming. This paradigm enable to "graft" traces into the evaluated system kernel since the first phase of system development life-cycles (Delannay, 2003). In other words, the interactive system evaluation is taken into consideration since the first development phase.

As mentionned previously, UI evaluation is generally supported only in the latest phase of the interactive system developpement cycle (e.g. testing phase in the Waterfall systems development life-cycle (Larman & Basili, 2003)). Then, many designers neglect user interface evaluation cause to hardware and time constraints.
The major motivation of the present work is to simplify user interface evaluation process. In addition, we intend to automate the evaluation process in order to provide more reliable results. In this paper, we propose to automate the evaluation process. Then, we intend to adopt a user interface evaluation approach. This evaluation is based on the inspection of the UI usability by exploiting ergonomic guidelines.

In this article, we are especially interested in tools that validate the ergonomic guidelines in the user interface evaluated. This interest is due to the fact that such evaluation is not costly according to other evaluation techniques. In addition to that, it is simple to establish and to obtain reliable results. Section 2 presents the state of the art for UI evaluation tools. Section 3 proposes our widgets dedicated to UI evaluation. They can be seen as a global tool for automated ergonomic guideline validation during the interface design process. Section 4 applies our approach to a network supervision system. Section 5 reports the results obtained and discusses our approach. Section 6 concludes the article and proposes perspectives for future research work.

2 Tools based on ergonomic guidelines validation for UI evaluation

Interaction devices are currently omnipresent in all domains. They are various and different according to many aspects (screen size, support medium, etc.). Among the interaction devices we can cite: PC, Samartphones, tablets, etc. With such devices, users can access the information wherever and whenever they want (Bacha, Oliveira, & Abed, 2011). This device diversity poses new challenges for UI evaluation. Therefore, the evaluation tools are presented in the following three main categories, which are related to the interface on which the interactive system operates (Figure 1):

- *WIMP[3]UI evaluation tools*: this category lists all the tools allowing the evaluation of WIMP user interfaces. These user interfaces operate generally on personal computers. The interaction between the interface and the user is mainly based on the use of mouse, screen and keyboard. In this category, there are not many tools. In fact, it is difficult to evaluate the ergonomic quality of such interfaces due to the fact that they are implemented through different programming languages. In addition, their source code is not often available to the evaluator. This is the reason why evaluating such systems mostly consists of integrating specific mechanisms and techniques (e.g., MESIA electronic informer (Trabelsi, Ezzedine, & Kolski, 2009) and questionnaire

[3] WIMP is the acronym for Windows, Icons, Menus and Pointing devices. WIMP user interfaces are the traditional user interfaces in which the interaction is based on the mouse and the keyboard.

exploitation[4] (van Velsen, van der Geest, & Klaassen, 2011)) to collect information for the evaluation. This information is analyzed to determine the ergonomic quality of the user interface and/or to detect the interface's ergonomic inconsistencies.

- *Web UI (or WUI)[5] evaluation tools*: this category includes the majority of the existing evaluation tools. It is dedicated for evaluating web pages. It is easier to evaluate the web pages' ergonomic quality than that of a WIMP interface. Generally, the evaluator has access to the HTML code in order to identify the graphic control attribute values for the evaluation. Therefore, the evaluation principle generally lies in the inspection of the conformity of the interface, according to guidelines set (e.g., ReWeb and TestWeb (Ricca & Tonella, 2001), AccessEnable (Brinck et al., 2002), EvalAccess (Abascal, Arue, Farjado, & Garay, 2006)).
- *Mobile UI evaluation tools*: nowadays, interactive systems operating on mobile phones, tablets and personal digital assistant terminals are evolving exponentially. Numerous applications are increasingly available with the iPhone, Android, and Mobile Windows (Rogers et al., 2011), for example (Monk, Carroll, Parker, & Blythe, 2004). Nevertheless, these system evaluation techniques and tools are rather rare. For instance, Lift Machine (Usablenet, 2004) evaluates Black Berry[6] terminal application interfaces.

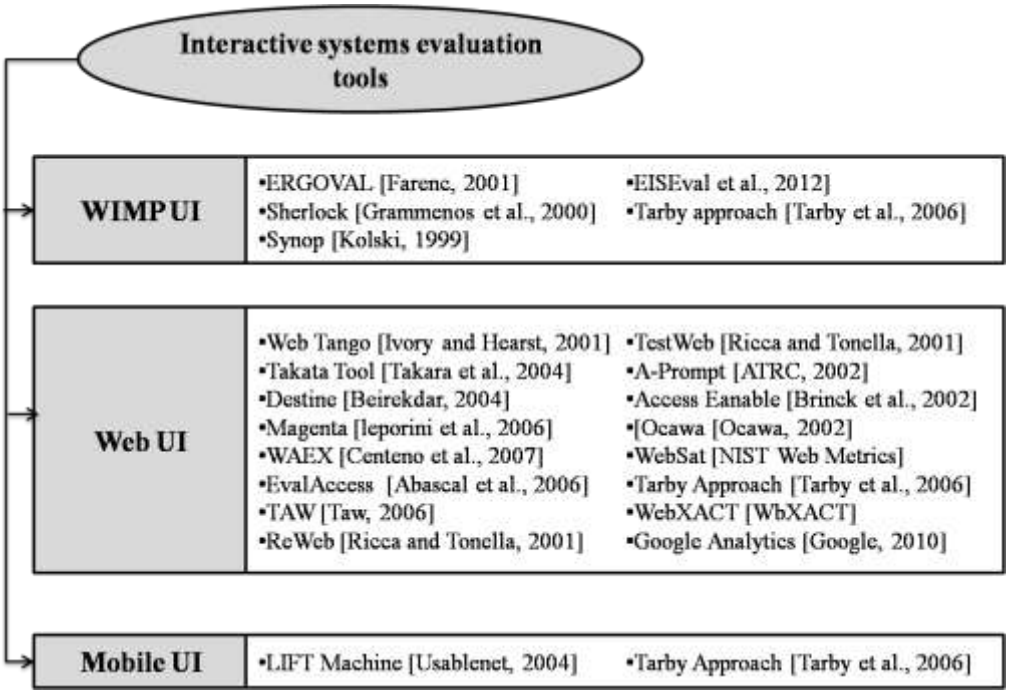

Figure 1: Classification of the tools for user interface evaluation

Table 1 lists representative evaluation tools, presenting some features for each tool:

- Acquisition: the technique to acquire data for the evaluation process (e.g., source code parser, textual description, questionnaire, electronic informer, and log file); the information can be captured automatically or manually.
- The evaluated user interface: WUI, WIMP or Mobile interfaces.
- Provided service: non-respected ergonomic guidelines and/or UI correction suggestions.

[4] To insure usability tests, there are three types of questions: pre-test questions, post-task questions and post-test questions (Sauro & Dumas, 2009).

[5] WUI is the acronym for Web User Interface. They are the user interfaces specifically for web pages, and they are used through the internet browser.

[6] http://us.blackberry.com/

- Evaluation type: static (user-interface graphic control attributes) and/or dynamic (user-interface interaction).
- Design phase: specification, design, implementation or final system testing.
- The inspected quality factor: accessibility, utility and usability.
- Automation: according to the evaluation process phases introduced by Ivory and Hearst (2001), we distinguish three phases:
  - Acquire the necessary data for the evaluation process,
  - Analyze the acquired data, and
  - Critique the user interface using the analyzed data to develop suggestions.

  Every automation phase can be done manually (M), semi-automatically (S) or automatically (A).
- Flexibility: whether or not the evaluation tool allows the evaluator to select the guidelines to be evaluated and to add new guidelines to EG database.
- The type of the evaluation tool: web site or software.
- Contributor: the user, the evaluator and/or the designer.

Table 1 lists some of the existing evaluation tools. These tools do not evaluate different types of UIs. For example, they evaluate only Web or WIMP user interfaces. The table above illustrates most of the user interface evaluation tools (16 tools from 20) evaluate WUI. Although the mobile applications are increasingly widespread, Mobile user interface evaluation tools are rare (only one tool from 20). In addition, the existing tools are applied during the last phase of the system development life cycle: the testing phase (in the waterfall systems development life-cycle (Larman & Basili, 2003)). Tools proposing an early evaluation are few (e.g., THEA (Pocock et al., 2001) and the Tarby approach[7] (Tarby et al., 2008)). Most of the tools in Table 1 do not provide an automated evaluation process. As seen in Table 1, 13 out of 20 tools do not propose automatic critiques; either S (semi-automatically) or M (manually), can be found. Then, the evaluation is done manually during the acquisition, analysis and critique.

Most of the evaluation tools in Table 1 propose only non-respected ergonomic guidelines as evaluation results. Some tools propose the graphic elements that do not correspond to inspected ergonomic guidelines. They do not generally correct the user interface automatically. However, they propose suggestions to improve the evaluated interface. In addition, the evaluation process is not easy to set up in the tools presented in Table 1. In fact, they require a good preparation of the UI evaluation and specific knowledge of the tools for the evaluation process.

[7] In the first phase of the interactive system development cycle, this approach grafts use-based features on the functional kernel, thus facilitating the evaluation phase. The approach is based on aspect-oriented programming and tasks.

Tools			Web UI evaluation														WIMP UI					Mobile UI
			Testweb	Takaka	A Prompt	Access Enabe	HTML Toolbox	Lift Machine	Taw	Bobby	Magenta	Destine	Waex	Hyper AT	Ocawa	Ergoval	EISEval	Sherlock	THEA	MESIA	Tarby approach[8]	Access Enable
Input	Acquisition	Parser	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X				X
		Textual Description																X				
		Questionnaire																X	X			
		Electronic Informer															X			X		
		Log file												X							X	X
Output	Provided service	Non-respected EG	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X				X
		Correction suggestions						X											X			
	Evaluation type	Static	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X			X
		Dynamic	X											X			X	X	X	X	X	
Design phase	Specification																		X			
	Design																					
	Implementation																					
	Final system testing		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Quality factor	Accessibility			X	X	X		X	X	X		X	X		X							X
	Utility								X								X			X	X	
	Usability		X				X	X		X	X			X	X	X	X	X	X	X	X	X	X
Automation	Acquisition		S	A	A	A	A	A	A	A	A	A	A	A	A	A	A	S	M	A	A	A
	Analysis		A	A	A	A	A	A	A	A	S	A	A	A	A	A	A	A	A	A	A	A
	Critiques		M	M	M	A	A	A	A	M	M	M	A	M	A	M	M	S	M	M	M	A
Flexibility	EG selection			X	X	X		X	X	X	X	X	X		X	X		X				X
	EG addition			X					X		X	X	X		X	X		X				X
Tool type	Web site				X				X				X		X							X
	Software		X	X	X		X	X	X	X	X	X		X		X	X	X	X	X	X	
Contributor	User											X		X			X	X		X	X	
	Evaluator		X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	Designer															X		X			X	

Table 1: List of the existing user interface evaluation tools

[8] This approach can be applied for WIMP, Web and Mobile user interfaces.

3 Proposition: Widgets dedicated to user interface evaluation

Evaluating the user interface can be defined as the validation of the user interface's conformity to ergonomic guidelines (Abascal et al., 2006; Beirekdar, Vanderdonckt, Noirhomme, 2002). Based on this definition, our approach evaluates a set of ergonomic guidelines, which are integrated into the widgets that constitute the user interface. This evaluation is made locally by the widget. In other words, our approach exploits a graphic interface widget set. These widgets are able to self-evaluate according to the predefined guidelines.

3.1 General presentation of our approach

Our approach is composed of three widget categories. Each category is dedicated to each UI type: WIMP, Web and Mobile UI. Each category is encapsulated into a DLL[9] file, thus making their exploitation easier. The objective of these widgets is to provide self-evaluation according to ergonomic guidelines. These guidelines are defined in advance by the evaluator. The originality of this approach lies in the coupling between the design phase and the evaluation phase.

The proposed evaluation process is automated during the three evaluation levels: acquisition, analysis and critiques (Section 2). Widget use is intended for WYSIWYG[10] programming environments. The proposed widgets are mostly used to aid the evaluator to evaluate usability which is an interactive system's ease of use in order to execute well-defined tasks; it guarantees intuitive handling and learnability, as well as support for using the graphic user interface.

This approach is classified under *Evaluation through expert analysis techniques* (Dix et al., 2003). As shown in Figure 2, it requires three contributors: a programmer, a designer and an evaluator. The designer has to conceive the interactive system's graphic interface. The programmer has to implement the personalized widgets. The evaluator has to specify and to select the guideline to use for the evaluation. The evaluation is based on the interface presentation according to ergonomic guidelines. It detects aspects related to these guidelines in the user interface.
The proposed widgets propose, as evaluation report, two reports:

- The first report informs the designer about ergonomic inconsistencies with specified guidelines. This report shows the widget aspects that do not correspond to the specified guidelines and contains suggestions to solve the ergonomic inconsistencies of the widget.
- The second report is a PDF file, which contains the ergonomic inconsistencies of the widgets and recommendations for improving the interface. It includes the different widgets notifications.

In other words, the first report is specific to a widget, while the second one concerns the whole user interface that was evaluated.

Figure 2 illustrates the proposed evaluation process, which revolves around two major stages. First, the evaluator selects EG for the evaluation. Then, the evaluator formalizes and defines the guidelines for the evaluation process during the specification phase (in the sense of the requirement phase in the waterfall systems development life-cycle (Larman & Basili, 2003)). These guidelines are saved into XML files (a file per guideline). Each XML file is created with a dedicated interface (Figure 5).

[9] Dynamic Link Library: a format a file used by Windows operating system. It is used to contain library used by programs.
[10] WYSIWYG is the acronym for "What you see is what you get". This acronym is used to indicate development environments that allow composing user interfaces visually.

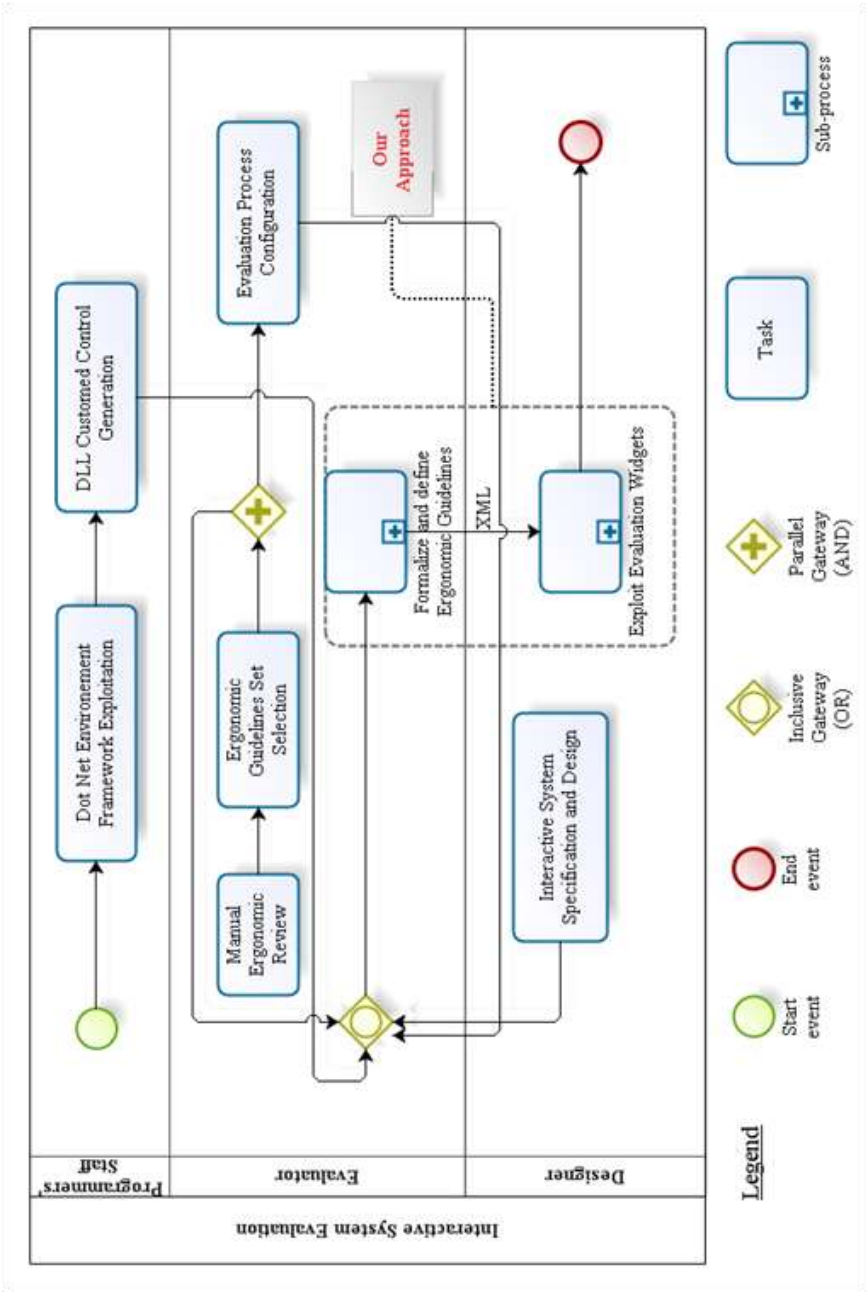

Figure 2: The general functioning of the evaluation process modeled through BPMN notations

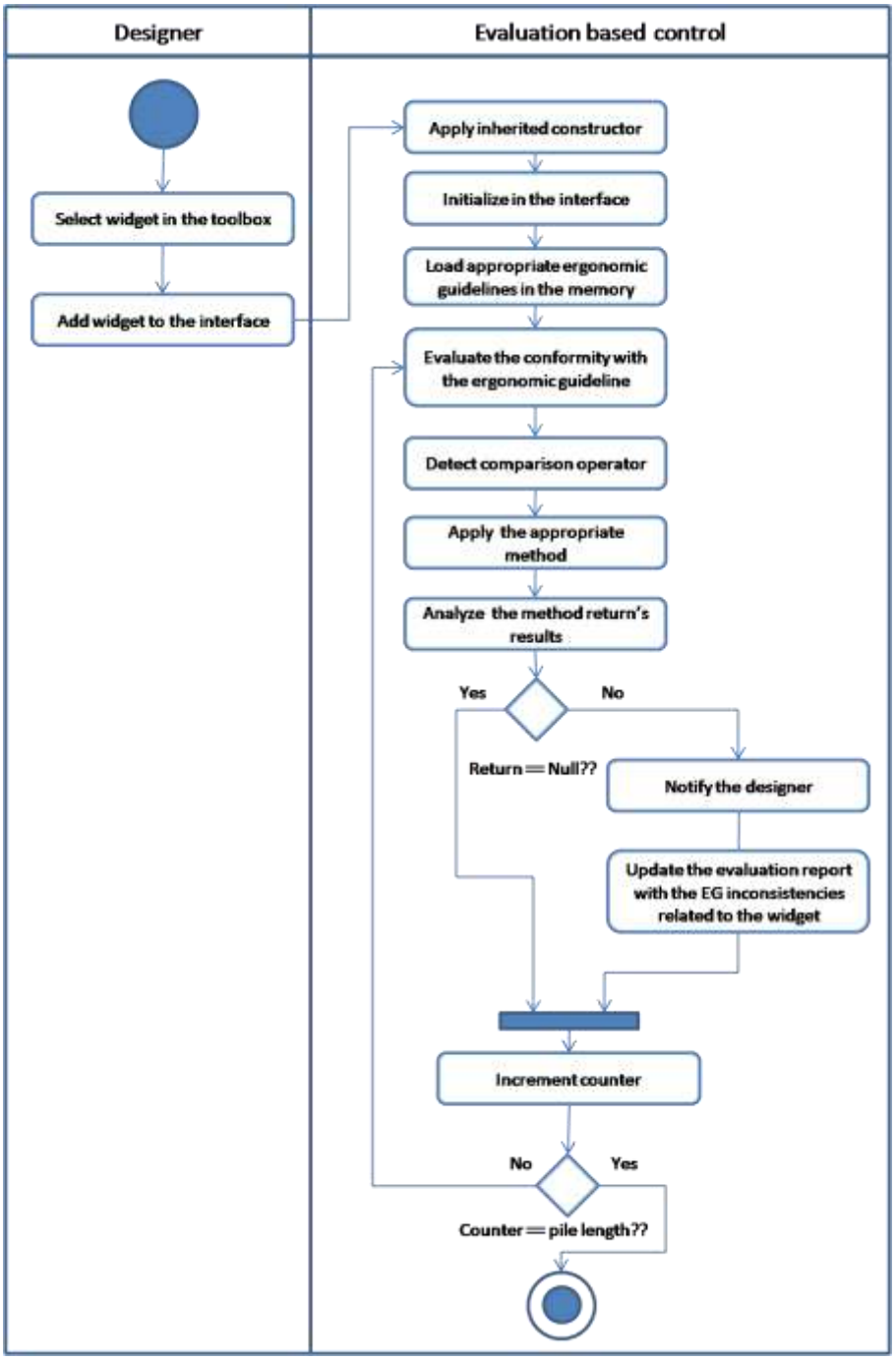

Figure 3: Activity diagram for the widget self-evaluation process

Figure 3 illustrates the actions performed by the widget during its creation. First of all, it initializes itself via an inherited constructor from the widget library provided by development environment. Then, the widget evaluates its conformity to specified guidelines set. As mentioned earlier, this set, specified by the evaluator, is appropriate for the interface type (i.e., WIMP, WUI or Mobile interfaces). Finally, the widget notifies the evaluator about ergonomic inconsistencies; this notification contains the non-respected guidelines and improvement suggestions, Figure 4. If the widget is coherent with the guidelines set, it informs the designer that there are no inconsistencies according to selected EG. Then, the widget gives the global report of the inconsistencies detected related to the specified guidelines and suggestions for improvement.

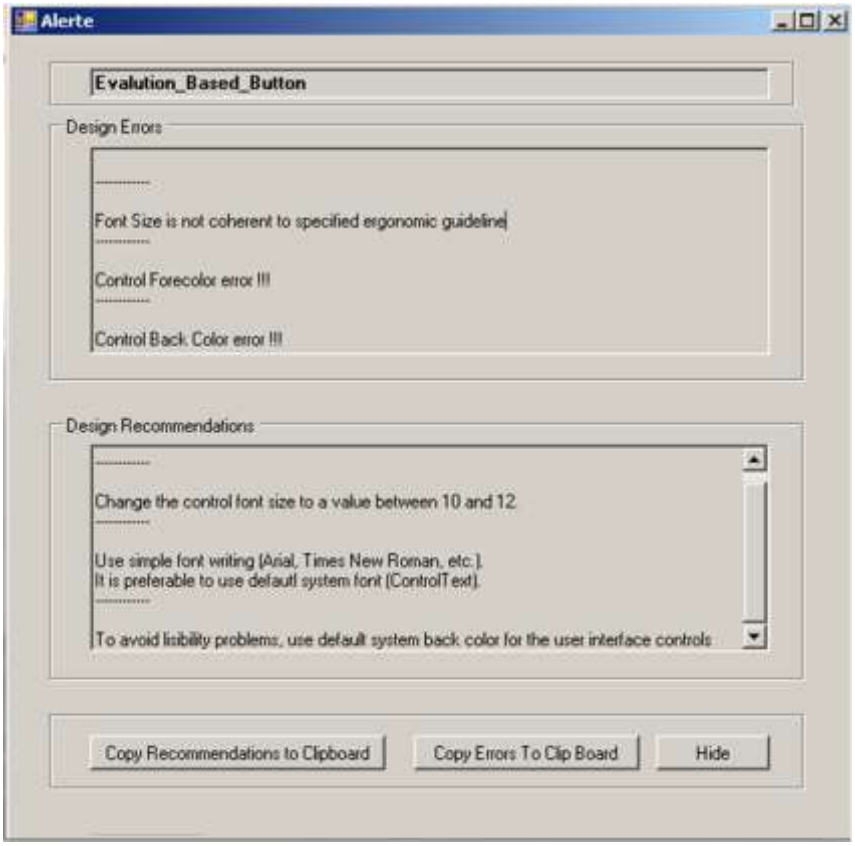

Figure 4. Notification Example

3.2 Widgets dedicated to early user interface evaluation

The proposed widgets appear to be similar to those proposed by the WYSIWYG development environments. In fact, they deal with the same functions. Shown on the widget toolbar, they can be used according to the Drag and Drop principle (Figure 12). Although, there is no apparent specificity to the designer. These widgets are endowed with additional mechanisms to evaluate their conformity according to the ergonomic guidelines set. The proposed widgets are separated into three categories: WIMP interface design, Web UI design, and Mobile UI design. Each category is encapsulated in a DLL file, which can be used to insert these widgets in the widget toolbar. In this paper, our widgets are intended for *"MS Visual Studio 2010"* development environment. It is possible to extend these categories to other environments (*"MS Borland C ++", "Eclipse",* etc.).

The pseudo-code below illustrates the self-evaluation process with the proposed widgets (Figure 5). First, the widget initializes itself on the graphic user interface using the inherited constructor. Then, it loads the ergonomic guidelines related to its type into a queue. Next, it analyzes its conformity to the guidelines according to the logical and arithmetic operator type used (e.g., superior, inferior, equal, different). Each operator is associated to a method. The widget appeals to the appropriate method by giving the attributes and guideline values as an argument. At the end of the queue parsing, the widget notifies the designer of the ergonomic guideline inconsistencies and saves these inconsistencies in the evaluation report (PDF file). Note that a guideline can be applied for more than a widget (for instance can be applied for textbox, label and button. This guideline is defined only one time and the associated widgets are mentioned in the "*Component Tag*").

```
General Evaluation widget algorithm
  Input:   XML files for the ergonomic guidelines
           Integrate the widgets for the design environment
  Output:  Conformity notification
           Evaluation report
Begin

  Apply inherited constructor /*As the proposed controls inherit from the IDE, they apply the inherited constructor to provide the classical controls features*/
  Initialize the widget on the user interface /*The control is drawn by itself on the designed interface*/
  Read XML files /* Parse the different XML files containing the ergonomic guidelines*/
  Load the XML file into an array list /*In the case that the guideline (expressed through XML file) is applied for the control type, this guideline is loaded into the memory*/
  Array list parse /*Parse the array list and apply the method associated to the guideline operator*/
  {  Load element(counter)
     While(counter<=arraylist.count) /*Parse the array list containing the guidelines to be inspected*/
        Loop
        Switch(operator) /* Apply the appropriated method according to the guideline operator */
          Case "Inferior": inferior(arraylist(i)); break;
          Case "Superior": superior(arraylist(i)); break;
          Case "Equal": equal(arraylist(i)); break;
          …
        Counter++
       End loop}
  Notification(error, recommendation).show(); /*Display the detected design error to the designer */
  Save(error, recommendation, report); /* Add the detected design error to the inspection report */

End
```

Figure 5: The pseudo-code of the widget dedicated to user interface evaluation

The operating principle of these widgets is described below. Once created using the Drag and Drop, the personalized widget launches the inherited constructor from the original class, proposed by the development environment framework. Then, it traces its shape on the interface. Next, it selects ergonomic guidelines, which are associated with its type (e.g., button, text field, checkbox). It extracts its attribute values to develop a comparison, which gives information about the widget's conformity to ergonomic guidelines (Figure 3).

3.3 Ergonomic guideline modeling

According to Vanderdonckt (1999), an ergonomic guideline is a design and/or evaluation principle to be observed to obtain and/or guarantee an ergonomic interface. Generally, it comes from other disciplines, such as software engineering, or from observing or studying interactive system users.

They are usually expressed in natural language to guide the designer and/or the evaluator to obtain useful, accessible and usable interfaces.

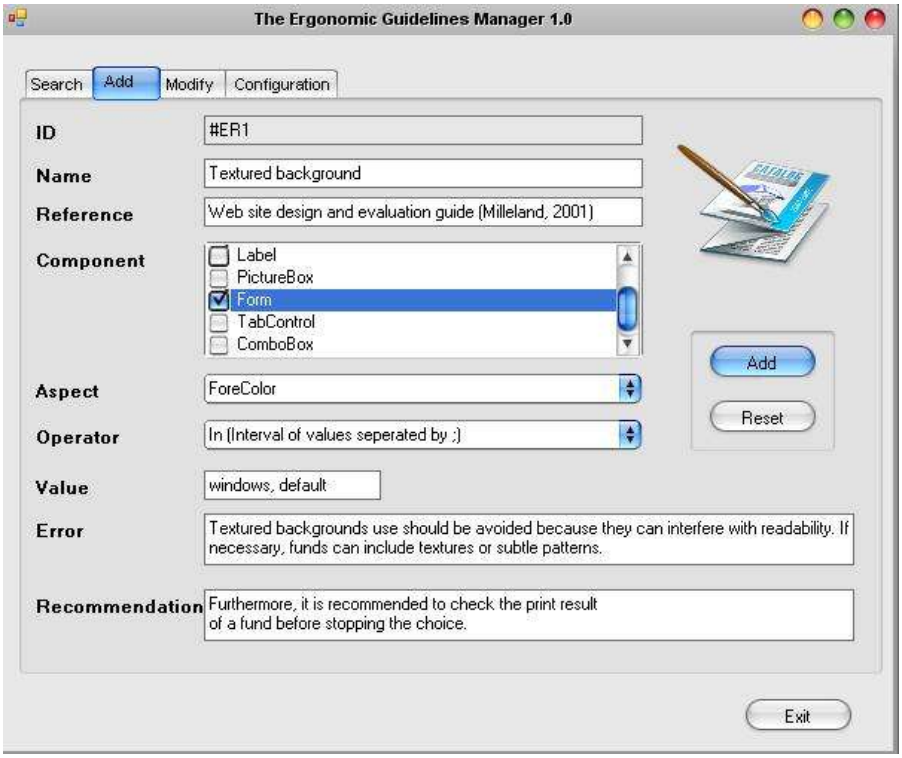

Figure 6. The Ergonomic Guidelines Manager

The proposed tool (Figure 6), called *the Ergonomic Guidelines Manager*, defines standardized guidelines so that these guidelines can be exploited for the UI evaluation. This tool allows:

- Consulting the saved ergonomic guidelines (*Search* tab) – The search can be done via the guideline identifier, name or reference.
- Adding new guideline (*Add* tab) – The guideline identifier is automatically generated by the system. The evaluator has to specify the name and the bibliographical reference, as well as all the widgets type to which the guideline can be applied. The tool proposes a widget list to the evaluator. In addition, it should express the guidelines through logical operators (e.g., equal, superior, inferior, between, equal, different, different from the group) and widget attributes (e.g., title, police, size, color, background). Then, the generated inconsistencies and correction suggestions have to be specified.
- Modifying the existing guidelines (*Modify* tab) – The guidelines saved in XML files can be modified, except for the guideline identifier.
- Configuring the Ergonomic Guidelines Manager (*Configuration* tab) – The path for saving evaluation reports and XML file must be specified (Figure 6).

Let us take the guideline example: " *An icon is a graphic that takes up a small portion of screen real estate and provides a quick, intuitive representation of an action, a status, or an app*." (Android, 2012). Figure 7 shows the XML representation of this guideline.

```
<Style>
  <EG_ID>#ER2</EG_ID>
  <EG_Name>Icons</EG_Name>
  <EG_Widgets>Button</EG_Widgets>
  <EG_Aspect>Icons</EG_Aspect>
  <EG_Operator>Is Different from; Is Not Empty</EG_Operator>
  <EG_Value1>"Null"10</EG_Value1>
  <EG_Value2> </EG_Value2>
  <EG_Error>Icons representation</EG_Error>
  <EG_Recommendation> An icon is a graphic that takes up a small portion of screen real estate and provides a quick, intuitive representation of an action, a status, or an app.</EG_Recommendation>
  <EG_Density>30</EG_Density>
</Style>
```

Figure 7: Example of a guideline expressed in XML notation

Another example is: *"Keep it brief: Use short phrases with simple words. People are likely to skip sentences if they're long*" (Android, 2012). Then we estimate the text label should not exceed 30 characters per label. Then, this guideline is modelled as follows, Figure 8:

```
<Style>
  <EG_ID>#ER13</EG_ID>
  <EG_Name>Keep it brief</EG_Name>
  <EG_Widgets>Label;</EG_Widgets>
  <EG_Aspect>Text.Length</EG_Aspect>
  <EG_Operator>Inferior</EG_Operator>
  <EG_Value1>30 </EG_Value1>
  <EG_Value2> </EG_Value2>
  <EG_Error>Much colors used in the user interface</EG_Error>
  <EG_Recommendation> Use short phrases with simple words. People are likely to skip sentences if they're long" </EG_Recommendation>
  <EG_Density></EG_Density>
</Style>
```

Figure 8: A second example of a guideline expressed in XML notation

Another example is: "*Given the unpredictability of colour screens and users, the choice can be very complicated. The colour is often best used to highlight key information. In general, do not use more than three primary colours for information*" (Watzman, 2002). This example is modelled as follows, Figure 9.

```
<Style>
  <EG_ID>#ER8</EG_ID>
  <EG_Name>Color_Number</EG_Name>
  <EG_Widgets>Button;TextBox;
RadioButton;ComboBox;Label</EG_Widgets>
  <EG_Aspect>Font.Color.Count</EG_Aspect>
  <EG_Operator>Inferior</EG_Operator>
  <EG_Value1>4 </EG_Value1>
  <EG_Value2> </EG_Value2>
  <EG_Error>Much colors used in the user interface</EG_Error>
  <EG_Recommendation> The colour is often best used to highlight key
information. In general, do not use more than three primary colours for
information. </EG_Recommendation>
  <EG_Density></EG_Density>
</Style>
```

Figure 9: A third example of a guideline expressed in XML notation

The EG should be contextualized, adequatelly interpreted then unambiguously specified and structured to be "quantifiable" and then suitable for being used with evaluation widgets. Once contextualized, the EG have to be defined using a formal language. Typically, they are expressed in ergonomic manuals in natural language, making exploiting them rather difficult. The EG exploitation remains at their contextual interpretations. Many languages are proposed for defining ergonomic guidelines (e.g., Guideline Definition Language (GDL) (Beirekdar et al., 2002), Guideline Abstraction Language (GAL) (Leporini et al., 2004), Unified Guideline Language (UGL) (Arrue, Vigo, Aizpurua, & Abascal, 2007)). Therefore, many ergonomic guidelines cannot be expressed. These developed languages are complicated and demand special tools for using them. Arrue et al. (2007) proposed UGL, which is a specific language for better guideline management. They also proposed a tool for modeling guidelines, which is dedicated for evaluating web site accessibility (Takata, Nakamura, & Seki, 2004). The guideline definition languages cited are based on the XML notations for reliability and simplicity. They are dedicated for evaluating web sites.

In our approach, we opted for a simpler guideline model (Figure 10). Our guideline modeling process consists of choosing the guideline to be considered for the design or evaluation phase. Second, the guideline's graphic aspect[11] (e.g., font, size, color, dimension) has to be specified. Third, the widget type associated to the guideline has to be selected. Fourth, the guideline is expressed through the arithmetical (e.g., superior, inferior, equal) and logical (e.g., and, or) operators[12]. Finally, the guideline is associated with the engendered inconsistency and the suggestions for improvements. The guideline is saved into an XML file (Figure 7).

[11] Note that a guideline can deal with more than one aspect (for example font colour and control size), it is defined through two distinct guidelines (one aspect per guideline).

[12] Like related aspects, the guideline can support only one operator by guideline. Thus, it is not possible to combine between several operators to define one guideline.

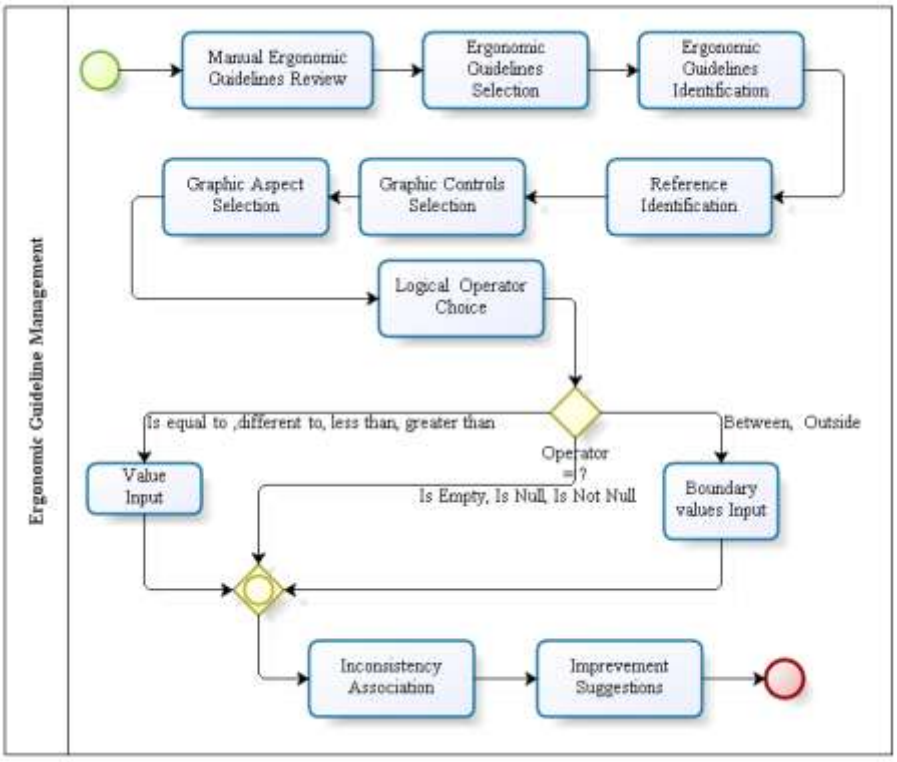

Figure 10: The process of ergonomic guideline definition into XML file

3.4 Exploitation of ergonomic guidelines by the proposed widgets for the UI evaluation process

For its self-evaluation, the widget goes through the guidelines selected by the evaluator. It duplicates in its memory the guidelines in which the guideline type appears with < *EG_widgets* > tag in the XML file (Figure 7). Then, the widget evaluates its conformity according to these guidelines. For every guideline, the widget identifies the selected operator (e.g., superior, inferior). A procedure corresponds to each operator. As inputs, the widget provides its attributes values and the recommended guideline values for the argument. Every time an inconsistency is detected, the character chains, "*recomm*" and "*error*", are furnished by the detected design inconsistency and the improvement suggestions. At the end of self-evaluation process, the widget notifies the designer with these characters chains in order to inform him/her about the detected ergonomic inconsistency.

4 Experimental evaluation

In order to validate and improve the proposed early evaluation approach, an experimental evaluation is proposed in this section. It deals with a system dedicated for network supervision prototype, Figure 10. The prototypes are conceived using the proposed widgets.

4.1 Evaluated system : The IAS

The IAS[13] (**I**nformation **A**ssistance **S**ystem) is a system dedicated for the transportation network information presentation. It is used by network regulators. Its main aim is to inform human

[13] The IAS (Information Assistance System) is a cooperative project involving an industrial partner (Transvilles) and several research laboratories (LAGIS, LAMIH and INRETS). This project is sponsored by the Nord-Pas de Calais regional authorities and by the FEDER (Fonds Européen de Développement Régional – European fund for regional development).

regulators about different vehicles position in the transport network. In addition to this, it enables regulators to communicate with vehicles drivers and passengers via sending messages, Figure 11.

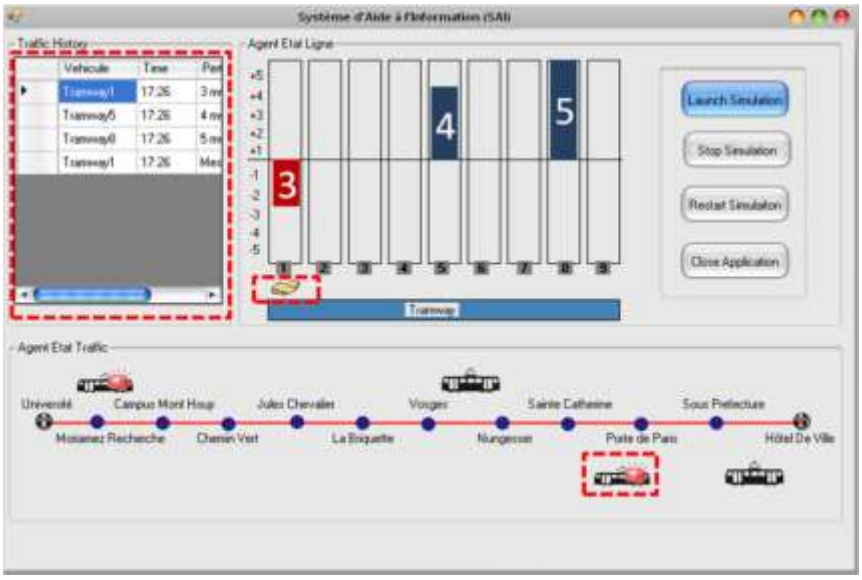

Figure 11. A prototype of IAS implemented using evaluation Widgets.

4.2 Design/Evaluation process

As mentioned earlier, the proposed evaluation process is coupled to the design phase. The evaluation is done through the interface design with the proposed evaluation widgets. Before proceeding to the interface design, evaluators have to select an EG set to take it into the consideration for the evaluation/design process. These EG are defined into XML files. The selected rules focus on: the writing size, the writing color, the writing font, image dimensions, graphic components size and menu item number.

Then, the designer compose graphically the user interface with the proposed widgets. Every time, a widget is added design errors and recommendation are displayed as a notification to the user. Once the interface is finalized, the designer disposes of a global report about the design error and improvement suggestion. Indeed, during the IAS design, the designer is informed by a set of recommendations proposed by the different UI components, Figure 12. The used widget for the IAS design are: button, label, picture box, text box and combo-box.

We defined an ergonomic guideline set for every user interface design or evaluation phase. Each set revolved around the information display:

- Character size, color and font;
- Size and number of the pictures and icons;
- Text length;
- Widget dimensions;
- Color number used;
- Global interface density; and
- Background color.

These guidelines were used to evaluate the usability of the interface. Our early evaluation verified the interfaces' conformity to the specified guidelines.

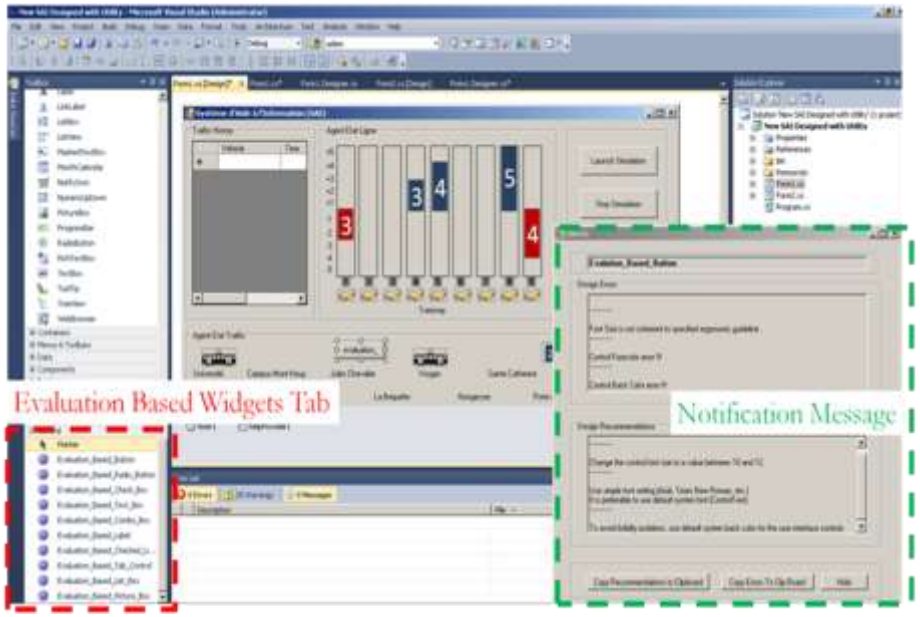

Figure 12. A screenshot of the user interface design/evaluation of the IAS using the evaluation based widgets.

4.3 Evaluation results

The IAS prototype evaluation did not raise major problems of usability. Indeed, design errors detected revolve mainly were about writing font adopted by the IDE "*MS Visual 2010"* (the used font is "*Microsoft Sans Serif*" while the associated RE recommends the use of the font used by the operating system). In addition, one of the selected EG recommends to use writing font size equal to ten (10) points. Meanwhile, the used font size is 8.25 points.

5 Results and discussion

In the IU design or evaluation phase, the proposed approach is an easy and effective method to assist user interface evaluators for early evaluation. It provides information about usability problems. Depending on the type of interactive system interface, an ergonomic guideline can be differently interpreted. One of our approach's advantages is the notification provided to the designer concerning the detected ergonomic inconsistancies.

In our approach, the UI evaluation is established during the design phase which makes it possible to save time and resources. Indeed, ergonomic inconsistencies are detected in the early stages of systems development life-cycle. As expressed in §1, Nielsen (1994) thinks that it is 100 times cheaper to correct errors during the first design phase than the last phase. The proposed evaluation provides design errors and improvement suggestions list. Although the evaluator can evaluate the conformity to the guidelines, he/she cannot evaluate the ergonomic quality. Our widgets do not indicate the quality of the user interface evaluated.

Compared to existing user interface evaluation techniques, our approach is easy to apply during earliest phases of systems development life-cycle: the design phase (in the case of the waterfall Systems development life-cycle (Larman and Basili, 2003)). As shown in Table 1, most of the tools are applied during the evaluation phase. Only one technique, THEA (Pocock et al., 2001), evaluates in the design phase. In addition, our approach can be applied to Web, WIMP and Mobile user interfaces. The evaluation process consists on detecting ergonomic inconsistencies in the evaluated UI. Furthermore, this approach is not limited to ergonomic guidelines set for evaluating the quality

ergonomic of the interactive systems; the guidelines are defined into XML files. Note that the supported ergonomic guideline for user evaluation are simple one that can be defined through the graphical interface controls and can be defined through the proposed logical and mathematical operators. For instance the guideline : "*Controls should allow individual users ease of access to media components that serve their individual needs.*" ISO/DIS 14915-2 (ISO 14915-2, 2001). can not be supported in the proposed evaluation process through the evaluation widgets.

Table 2 compares the proposed approach to those presented in Table 1. Our approach makes it possible to save time in the evaluation of the user interface. The evaluation process through the proposed approach is totally automated, with acquisition, analysis and critique phases. The main advantage, compared to existing approaches, is the fact that it is applied in early stages of system development. In addition to that, unlike most of the tools, the proposed approach do not hard code the guidelines into the evaluation engine. They are coded externally of the evaluation engine. As the guidelines are coded externally as XML files, they can easily be modified. The proposed approach focuses on the static presentation of a user interface not like THEA technique that is dedicated to asking questions and exploring interactive system designs to know how a device functions in a scenario. The proposed approach is used independently to use scenarios. Another aspect remains in the fact that the proposed approach can be applied for WIMP, Web and Mobile user interfaces.

Tools			Existing Tools (20 Tools in total)	Our approach
Input	Acquisition	Parser	16	X
		Textual Description	1	
		Questionnaire	2	
		Electronic Informer	2	
		Log file	2	
	Evaluated User Interface	Web UI	14	X
		WIMP UI	5	X
		Mobile UI	1	X
Output	Provided service	Non-respected EG	16	X
		Correction suggestions	2	Perspective
	Evaluation Type	Static	17	X
		Dynamic	7	
Design phase	Specification		1	
	Design		1	X
	Implementation		0	
	Final system testing		20	
Quality factor	Accessibility		11	X
	Utility		4	
	Usability		14	X
Automation	Acquisition		17 Automatically	A
	Analysis		19 Automatically	A
	Critiques		7 Automatically	A
Flexibility	EG selection		13	A
	EG addition		9	A
Tool type	Web site		5	
	Software		16	X
Contributor	User		16	
	Evaluator		19	X
	Designer		3	X

Table 2: Our approach compared with the existing tools shown in Table 1

6 Conclusions

This paper presents an approach for the early UI evaluation. The originality of this research lies in imbricating evaluation into the widgets. This evaluation is based on the widgets checking conformity to a set of ergonomic guidelines. The advantage of our approach is its ease of use during the design or evaluation phase. In addition, it integrates new ergonomic guidelines without touching the widget source code. These widgets can be used for WIMP, Web and Mobile UI. The proposed approach does not require user participation into the evaluation process. It belongs to the category of tools related the evaluation through analysis techniques (in the sense of (Dix et al., 2003).

As a perspective for future research, we will integrate more widgets in the evaluation process (in three categories). Our widgets were developed for studying the feasibility of our approach. In addition, we will improve the quality of the evaluation reports. Consequently, we will use report standards, such as RDL (Microsoft Corporation, 2009) and EARL (Word Wide Web Consortium, 2009), making the evaluation reports easier to manage and to understand. The evaluation report schould integrate graphs for a better understanding.

Our approach identifies only the ergonomic inconsistencies within the widgets. This evaluation is done locally at the widget level, evaluating the interface's conformity with the guidelines, widget by widget; it does not evaluate the whole interface. This can prove to be inadequate because the interface may contain ergonomic inconsistencies when the widgets are in conformity with the specified guidelines. Therefore, we suggest combining our approach with an approach permitting a dynamic evaluation of the interaction between the user and the interactive system.

One limitation of the proposed widgets is the fact that they support only basic features. We intend to develop the proposed widgets by taking into consideration their behaviour (they are activated or not, the associated events, etc.). In addition to that, we intend to integrate Artificial Intelligence into the proposed widgets to allow them to communicate and handle design problems and to support the evaluation of distributed interfaces (de la Guía, Penichet, Garrido, & Albertos, 2012).

As perpective, we intend also to extend the evaluation process to other systems development life-cycle phases (e.g. in the case of Waterfall systems development life-cycle : the implementation, verification and maintenance phases) in order to take into consideration more aspects for the evaluation and then to get better evaluation results.

We also intend to extend this approach to support the evaluation of other types of user interfaces such as: Post-WIMP and Distributed user interfaces (Tesoriero & Lozano, 2012; Lepreux, Kubicki, Kolski, & Caelen, 2012).

Acknowledgements

This research is partially financed by the International Campus on Safety and Intermodality in Transportation (CISIT), the Nord/Pas-de-Calais Region, the European Community, the Regional Delegation for Research and Technology, the Ministry of Higher Education and Research, and the CNRS.

References

Abascal, J., Arue, M., Farjado, I., & Garay, N. (2006). An expert-based usability evaluation of the EvalAccess web service. In R. Navarro-Prieto, & J. Lorés (Eds.), *HCI related papers of Interacción*, (pp. 1-17), Springer.

Android. (2012) User Interface Guide, Retrieved April 10th, 2012, from http://developer.android.com/design/get-started/principles.html.

Arrue, M., Vigo, M., Aizpurua, A., & Abascal, J. (2007). Accessibility Guidelines Management Framework. In C. Stephanidis (Ed.), *Universal Access in Human-Computer Interaction. Applications and Services* (pp. 3-10), Lecture Notes in Computer Science, Vol 4556, Springer Berlin Heidelberg.

Bacha, F., Oliveira, K., & Abed, M. (2011). Using Context Modeling and Domain Ontology in the Design of Personalized User Interface. *International Journal on Computer Science and Information Systems (IJCSIS)*, 6, 69-94,

Bardzell, J. (2011). Interaction Criticism: An Introduction to the Practice. *Interacting With Computers*, 23, 604-621.

Beirekdar, A., Vanderdonckt, J., Noirhomme, M. (2002). A Framework and a Language for Usability Automatic Evaluation of Web Sites by Static Analysis of HTML Source Code. In Proceedings of *4th Int. CADUI* (337-348), Valenciennes, France, Kluwer Academics Pub.

Boy, G.A. (2011). *The Handbook of Human-Machine Interaction, A Human-Centered Design Approach*. Florida Institute of Technology, USA, Florida Institute for Human and Machine Cognition, and NASA Kennedy Space Center, USA, ISBN: 978-0-7546-7580-8.

Brinck, T., Hermann, D., Minnebo, B., & Hakim, A. (2002). AccessEnable: A Tool for Evaluating Compliance with Accessibility Standards. In: *CHI'2002 Workshop on Automatically Evaluating the Usability of Web Sites*, Florida, USA.

Buxton, B. (2007). *Sketching User Experiences: Getting the Design Right and the Right Design*. (Interactive Technologies), 1st Edition, Morgan Kaufmann, ISBN-13: 978-0123740373.

de la Guía, E., Penichet, V.R., Garrido, J.E., & Albertos, F. (2012): Design and Evaluation of a Collaborative System That Supports Distributed User Interfaces, *International Journal of Human-Computer Interaction*, 28(11), 768-774.

Delannay, G. (2003). A generic traceability tool. Retrieved from http://www.info.fundp.ac.be/~pth/fundpdocs/gde.pdf, 2003

Dix, A., Finlay, J., Abowd, G., & Beale, R. (2003). *Human–Computer Interaction*. 3rd Edition. Prentice Hall, ISBN 0-13-046109-1.

Filippi, S., & Barattin, D. (2012). Generation, Adoption, and Tuning of Usability Evaluation Multimethods, *International Journal of Human-Computer Interaction*, 28(6), 406-422.

Folmer. E., & Bosch. J. ,2004). Architecting for usability: A survey, *Journal of Systems and Software*, 70 (1-2), 61-78.

ISO 14915-2. (2001). *DIS 14915-2, Software ergonomics for multimedia user interfaces - Part 2: Multimedia navigation and control*. International Standad Organisation.

Juristo, N., Moreno, A.M., & Sanchez-Segura, M.I. (2007). Analysing the impact of usability on software design, *Journal of Systems and Software*, 80(9), 1506-1516.

Grudin, J. (1992). Utility and usability: Research issues and development contexts. *Interacting with Computers*, 4(2), 209-217.

Hearst, M. A. (2009). *Search User Interface*, Press, Cambridge University.

Hvannberg, E., Law, E., & Lárusdóttir, M.: Heuristic Evaluation: Comparing Ways of Finding and Reporting Usability Problems. *Interacting With Computers*, 19(2), 225-240.

Ivory, M., & Hearst, M. (2001). The State of the Art in Automated Usability Evaluation of User Interfaces. *ACM Computing Surveys*, 33(4), 173-197.

Konstan, J. (2011). Tutorial / HCI for recommender systems: a tutorial. in 'Proceedings of the *16th international conference on Intelligent user interfaces*, ISBN 978-1-4503-0419-1, Palo Alto, CA, USA.

Kortum, P. (2009). *HCI Beyond the GUI: Design for Haptic, Speech, Olfactory, and Other Nontraditional Interfaces* (Interactive Technologies). Morgan Kaufmann; 1st edition, ISBN-13: 978-0123740175.

Larman, C., & Basili, V.R. (2003). Iterative and Incremental Development: A Brief History. *Computer*, 36(6), 47-56.

Leonidis, A., Antona, M., & Stephanidis, C. (2012). Rapid Prototyping of Adaptable User Interfaces, *International Journal of Human-Computer Interaction*, 28(4), 213-235.

Leporini, B., & Paternò, F. (2004). In-creasing Usability when Interacting through Screen Readers. *International Journal Universal Access in the Information Society (UAIS), special Issue on "Guidelines, standards, methods and processes for software accessibility"*, Springer Verlag, 3(1), 57-70.

Lepreux, S., Kubicki, S., Kolski, C., & Caelen, J. (2012). From Centralized interactive tabletops to Distributed surfaces: the Tangiget concept. *International Journal of Human-Computer Interaction*, 28(11), 709-721.

Mahatody, T., Sagar, M., & Kolski, C. (2010). State of the Art on the Cognitive Walkthrough Method, Its Variants and Evolutions. International Journal of Human-Computer Interaction, 26(8), 741-785.

Medvidovic, N. & Jakobac, V. (2005). Using Software Evolution to Focus Architectural Recovery. *Automated Software Engineering*, 13 (2), 225-256.

Microsoft Corporation, (2009). *Report Definition Language Specification*, Third Edition.

Monk, A.F., Carroll, J., Parker, S., & Blythe, M. (2004). Why are mobile phones annoying? *Behaviour and Information Technology*, 23, 33-42.

Nielsen, J. (1994). Heuristic evaluation. In J. Nielsen & R.L. Mack (Eds.), *Usability Inspection Methods* (pp. 25-62), John Wiley.

Pocock, S., Harrison, M., Wright, P., & Johnson, P. (2001). THEA – A technique for human error assessment early in design. Human-Computer Interaction: *INTERACT'01*, M. Hirose (Ed), pp. 247-254. IOS Pres.

Rafla, T., Robillard, P.N., & Desmarais, M. (2006). Investigating the impact of usability on software architecture through scenarios: A case study on Web systems, *Journal of Systems and Software*, 79 (3), 415-426.

Ricca, F., & Tonella, P. (2000). Web Site Analysis: Structure and Evolution. *Proc. 16th IEEE Int. Conf. on Software Maintenance (ICSM'00)*, 76-86.

Rogers, Y., Sharp, H., & Preece, J. (2011). *Interaction Design: beyond human-computer interaction*, 3rd edition, John Wiley, ISBN: 978-0470665763.

Salvendy, G., & Turley, L. (2002). Effectiveness of user testing and heuristic evaluation as a function of performance classification. *Behaviour & Information Technology*, 21(2), 137-143.

Savidis, A., & Stephanidis, C. (2006). Automated user interface engineering with a pattern reflecting programming language. *Automated Software Engineering*, 13(2), 303-339.

Sauro, J., & Dumas, J.S. (2009). Comparison of Three One-Question, Post-Task Usability Questionnaires. In Proceedings of the SIGCHI *Conference on Human Factors in Computing Systems*, CHI '09 (1599-1608), Boston, MA, USA, ACM, New York, NY, USA.

Sommerville, I. (2010). *Software Engineering*. 9th Edition, Addison-Wesley.

Takata, Y., Nakamura, T., & Seki, H. (2004). Accessibility Verification of WWW Documents by an Automatic Guideline Verification Tool. Proceedings of the *37th Annual Hawaii International Conference on System Sciences (HICSS'04)* - Track 4.

Tarby, J.C., Ezzedine, H., & Kolski, C. (2008). Prevision of evaluation by traces during the software design of interactive systems: two approaches compared. In A. Seffah, J. Vanderdonckt, & M. Desmarais (Ed.), *Human-Centered Software Engineering: Architectures and Models-Driven Integration*, Springer HCI Series, 257-276.

Tesoriero, R., & Lozano, M.D. (2012). Distributed User Interfaces: Applications and Challenges, *International Journal of Human-Computer Interaction*, 28(11), 697-699, DOI: 10.1080/10447318.2012.715048

Trabelsi, A., Ezzedine, H., & Kolski, C. (2009). Evaluation of agent-based interactive systems, application to an information assistance system: first results. In M. Sayed-mouchaweh (Ed.), *28th European Annual Conference on Human Decision-Making and Manual Widget*, Reims, 45-50, septembre, ISBN 978-2-915271-34-8.

Tran, C.D., Ezzedine, H., & Kolski, C. (2008). Evaluation of agent-based interactive systems: proposal of an electronic informer using Petri Nets. *Journal of Universal Computer Science*, 14(19), 3202-3216.

UsableNet Inc. (2004) LIFT for Dreamweaver Nielsen Norman Group edition. Retrieved April 10th, 2012, from http://www.usablenet.com/productsservices/lfdnng/lfdnng.html

Vanderdonckt, J. (1999) Development Milestones towards a Tool for Working with Guidelines. *Interacting With Computers*, 12(2), 81-118.

van Velsen, L., van der Geest, T., & Klaassen, R. (2011): Identifying Usability Issues for Personalization During Formative Evaluations: A Comparison of Three Methods, *International Journal of Human-Computer Interaction*, 27(7), 670-698.

Watzman, S. (2002). Visual design principles for usable interfaces. In A. Sears A & J.A. Jacko (Eds.), *The Human Computer Interaction Handbook*, CRC Press, 263 – 285.

Wharton, C., Bradford, J., Jeffries, J., & Franzke, M. (1992). Applying Cognitive Walkthroughs to more Complex User Interfaces: Experiences, Issues and Recommendations. *CHI '92*, 381–388.

Word Wide Web Consortium, (2009). *Evaluation And Report Language 1.0*. Retrieved April 10th, 2012, from http://www.w3.org/TR/EARL10/

Wright, P., Blythe, M., McCarthy, J., Gilroy, S., & Harrison, M.: User Experience and the Idea of Design. In Proceedings of the *12th international conference on Interactive Systems: Design, Specification, and Verification*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, SN - 978-3-540-34145-1, 1-14.

ABOUT THE AUTHORS

Selem CHARFI has obtained his Ph.D at the University of Valenciennes (France) in 2013. His research concerns human-computer interaction (HCI), agent-based architecture models of interactive systems, software engineering and HCI evaluation, with application to the supervision of transport systems. He is co-author of several papers in international conferences. He is involved in several research networks.

Abdelwaheb TRABELSI has obtained his Ph.D at the University of Valenciennes (France) in 2006. He is assistant professor in Computer Science at the University of Sfax (Tunisia) and member of the "Industrial management and decision-making support" LOGIC laboratory. He is involved in several research networks and projects. He is specialized in human-computer interaction and software engineering for interactive systems.

Houcine EZZEDINE has obtained his Ph.D in 1985. He is professor in Industrial Computer Science at the University of Valenciennes (France) and member of the "Human-Computer Interaction and Automated Reasoning" research group in the LAMIH. He is involved in several research networks, projects and associations. He is specialized in human-computer interaction and software engineering for interactive systems.

Christophe KOLSKI has obtained his Ph.D in 1989. He is specialized in human-computer interaction, software engineering for interactive system design and evaluation, adaptive UI. He is a professor in Computer Science at the University of Valenciennes and head of the “Human-Computer Interaction and Automated Reasoning” research group in the LAMIH. He is author or editor of several books, and author or co-author of many book chapters, papers in journals, communications in international congresses, and research reports in relation with industry.