Chapter

AUTOMATING GUIDELINES INSPECTION

From Web site Specification to Deployment

Joseph Xiong[1], Mouhamed Diouf[2], Christelle Farenc[1], Marco Winckler[1]
[1]*LIIHS-IRIT, Université Paul Sabatier*
118 Route de Narbone, 31062 Toulouse Cedex 4, France
Email: {xiong, farenc, winckler}@irit.fr
Phone: +33 5.61.55.63.59 – Fax : +33 5.61.55.62.58
[2] *L3A-LaBRI*
Domaine Universitaire, 351, cours de la Libération 33405 Talence Cedex, France
Email: diouf@labri.fr

Abstract: This work focuses on how we can improve automatic evaluation based on guidelines inspection throughout the life cycle of Web applications by mapping guideline concepts to different artifacts produced during the development process. In order to support such an evaluation approach, we present a tool for automated evaluation based on guidelines reviews.

Key words: automated usability evaluation, ergonomic inspection, usability, accessibility, Web sites, development process.

1. INTRODUCTION

Usability evaluation is a complex task that requires knowledge and expertise. For usability experts, most of the evaluation methods are "simple". For non-experts, however, evaluation methods are considered "complex". Usability guidelines may be one way to alleviate this lack of expertise. However, even experts experience difficulties in applying guidelines, at least in the format in which they are conflict with one another because there is a wide gap between the recommendation (ex. "*make the site consistent*") and its applications [6]. The SUE inspection method [8] helps to transfer the ergonomic knowledge by guiding novice evaluators through evaluation patterns (called abstract tasks) which tailor the manual inspection. However,

such as an approach requires a previous training to employ abstract tasks. In addition, the scope of the evaluation is limited by the coverage of predefined patterns.

In order to overcome these limitations of manual inspection, much effort has been devoted in the development of tools supporting the collection and the organization of guidelines [14] [15], guiding the inspection [1] and supporting automated guidelines inspection [1] [12]. One of the advantages of employing automated tools for guidelines inspection is that no extensive knowledge on ergonomics is required since most the technical details are embedded into the tools. The lack of experts to perform usability evaluations [12] and the high costs of other evaluation methods (e.g. user testing) make automated inspection of guidelines a suitable method for supporting frequent evaluations of Web applications.

Usability evaluation is not only important for identifying problems in the early phases of the development process but also for managing the frequent updates performed over web applications. According to the phase of the development life cycle different categories of guidelines can be employed for supporting design and/or evaluation of user interfaces [7]. In fact, in the over past years, several tools supporting automatic inspection of the HTML/CSS code of Web pages emerged as a valuable approach for improving usability of Web applications [1][3]. However, such tools only can be employed in late phases of development when a prototype of the user interface is available.

In our previous work we have investigated the guidelines verification on specific models (dialog models [9] and navigation models [17][18]) at particular phases of the life cycle and we have demonstrated the advantages of the guidelines verification over models in early phases of development. In the present work we propose a broader view for automated inspection of guidelines by inspecting all the artifacts produced at different phases throughout the life cycle of Web applications. We present a tool support which demonstrates our approach. The rest of this paper is organized as follows: Section 2 describes requirements for automated inspection of guidelines. Section 3 describes the general architecture of our proposal for automated evaluation using guidelines during the development process. Section 4 provides some examples which demonstrate our approach. Section 5 presents the set of tools dealing with implementation issues. Lastly, we present discussion and future work.

2. AUTOMATING GUIDELINES INSPECTION

The automation of guidelines verification is a complex process. First of all, it is needed to interpret high-level guidelines and to translate them into more concrete rules where each rule has a unique meaning. In the sequence, concrete rules should be mapped to user interfaces elements (e.g. navigation elements, text content, graphics, menus, etc) which can be used as input for automated tools. In addition, before running out with the automation, guidelines should be organized in the way they are useful at each phase of development process [11].

Hereafter, we describe the steps towards the automated inspection of guidelines. The set of guidelines used in this work is issued from W3C/WAI accessibility initiative[1] and the EvalWeb project [10].

2.1 Guidelines Interpretation

In order to be automated, guidelines have to be described in a precise and non-ambiguous way. Guidelines are expressed in natural language and an interpretation phase is often necessary to translate them into a computing language. Indeed, a rule can be more or less abstract [15] and is not appropriate to an automatic translation in all the cases. If so, rules should be decoded in one or more concrete rules. For example, the rule "*Provide accessible content for visual impaired users*" is too imprecise to be automatically implemented. So it should be interpreted in a more concrete way having a single meaning. For example: "*Provide alternative text for images"*.

Quite often, many concrete rules must be created in order to cover all possible facets a guideline might have with respect to the user interface content or widgets. For example, what does the rule "*Provide accessible content for visual impaired users"* mean for images, menus, buttons, animation, and so on?

The translation process might also creates a semantic distance between the original guidelines and the concrete rules created to support automated inspection. Considering the example above, the concrete guideline "*Provide alternative text for images"* only helps to check if a text description is associated to images; this can be considered as a kind of syntactic verification. Currently, there is no available tool supporting the verification of the semantics of alternative text and image's content, thus creating a gap between the rule and what is possible to verify concerning the rule.

[1] At http://www.w3.org/TR/WAI-WEBCONTENT/ (May 5th,1999)

2.2 Mapping Guidelines to User Interface Elements

After we have removed possible ambiguities in the guidelines, we must identify which user interface elements are concerned. For example, the following accessibility guideline: "*Provide alternative text for images*" will typically address the "IMG" tag and its "ALT" attribute embedded into the HTML. At this point we have two choices: rewrite the rule using a kind of '*algorithms'* processing such UI elements, or alternatively, formally describe ergonomic rule using specific temporal logics such as ACTL [4].

Several guidelines address HTML elements (e.g. images, links, paragraphs, etc.) and their attributes (e.g. link color, textual aid for images, table caption, etc.) that can be easily identified by analyzing the DOM[2] structure of HTML pages. However, many guidelines refer to abstract elements that are not formally described by HTML tags. For example, the evaluation of the following guideline: "*Include navigational aid pointers across to main sections*" can't be automated because there is no information concerning the elements "*navigational aid*" and "*main sections"*. We can generalize this situation as follows: the guidelines vocabulary used to describe abstract elements (e.g. "*navigational aid*" and "*main sections"*) cannot be automatically translated to the vocabulary (e.g. HTML tags) used to describe the artifacts (e.g. Web pages) being evaluated.

2.3 Development life cycle of Web applications

As mentioned before, our main goal is to apply automated inspection at early phase of the development process of the Web applications. So, we had to determine whether we can apply a guideline or not at each phase of the life cycle. Currently, there is no consensus on which phases of development are required, neither which life cycle better describes the development process of Web applications. Despite this, the life cycle for Web development can be generalized as an iterative process [12]. In our work, we follow an iterative design process which is made up of six phases as follow: 1) requirements engineering, 2) Site specification, 3) Site design, 4) Site development, 5) Site usage and evaluation, and 6) Maintenance. This life cycle, called in "O", is presented by Figure 1.

At each phase, several artifacts are produced. For example, the phase 2 (i.e. Site Specification) provides navigation models, data models, and so on, while the phase 3 (i.e. Site design) will provide templates describing the layout of pages; Web pages are produced during the phase 4 (i.e.

[2] http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109/ (January 9th, 2003)

implementation) and during the phase 5 we can collect reports concerning the usage metrics.

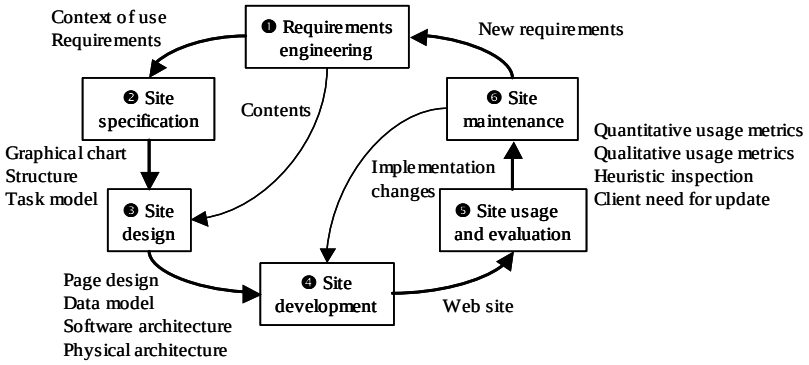

Figure 1 – General life cycle of Web applications development.

Artifacts describe a set of user interface elements on which we might apply ergonomic knowledge. However, the meaning and the applicability of each ergonomic rule can vary according the artifacts take into account; mostly because the ergonomic rules address user interface elements which are not described by the artifact. So, according to the artifact available at each step, a different set of guidelines can be applied. This is the basic assumption of our approach for automated evaluation of guidelines, which is presented in the next section.

2.4 Organizing Guidelines for Automated Evaluation

Not all guidelines can be fully automated. According to the degree of automation, we have classified our guidelines as follows:

- **Integrated**: it can be embedded into the design tools thus forcing the designer to respect it. Ex. "*Provide the size of images"*.
- **Automatic**: no human intervention is needed to assess the guideline. Ex. "*Check that all links can be navigated*".
- **Semi-Automatic**: the system can identify the appropriate elements over the user interface but it requires additional information to complete the evaluation. Ex. "*Provide meaningful label for links*".
- **Manual**: the system cannot determine which user interface elements are concerned by a guideline. Ex. *"Don't publish copyrighted material without permission of the owner*". So, the guideline cannot be automated.

The automation degree depends upon the appropriated mapping of guideline concepts and the artifacts elements available. It is worth noting that a same general guideline could be evaluated manually at a given step (e.g. design phase) and fully automated at the next step (e.g. implementation).

3. THE GENERAL ARCHITECTURE OF OUR PROPOSAL

Figure 2 briefly summarizes the general architecture of our approach. At the top, we have the life cycle of Web applications as describe in [12]. At each step, we produce some artifacts such as navigation models at the specification phase and templates at the design phase. For example, we produce *"navigation models"* artifacts at the step 2 (i.e. site specification), *"templates"* artifacts at the step 3 (i.e. site design), and so on. The artifacts can vary according to the methods and tools used to specify and create the design.

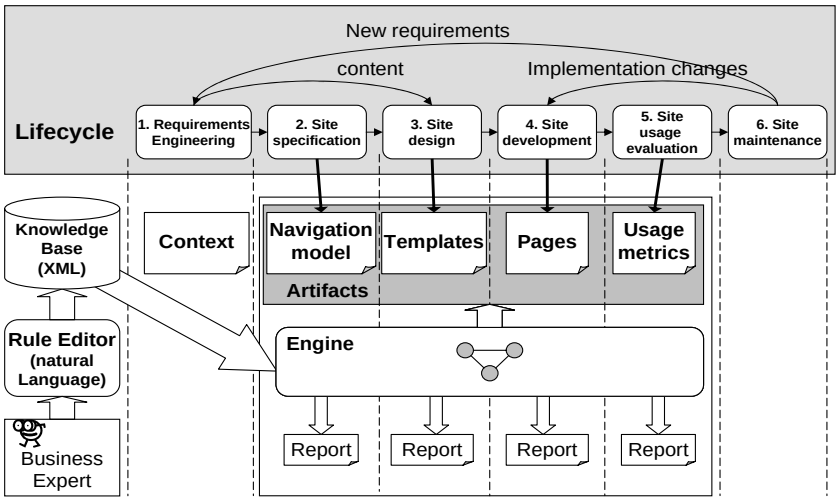

Figure 2 – General architecture for supporting automated guideline inspection throughout the development process of Web applications.

The artifacts are used as source of information to a rule engine enabling the automated processing of guidelines from a database. At the present, we don't have investigated how we could exploit artifacts produced at the phase of requirements engineering for supporting automated evaluation. So, we only are able to cover the phase of specification, site design, site development and site usage. The phase maintenance corresponds to the operation performed to solve problems. Our approach is supported by a guideline editor (at left side in Figure 2). The knowledge base stores the guidelines as well the corresponding mappings to the artifacts elements. A rule engine (i.e. namely Engine in Figure 2) uses the guidelines-artifacts mapping information stored in knowledge base to inspect in a seamless way all the artifacts.

In order to solve the mapping-problem between guidelines concepts and artifacts, we have defined a non-ambiguous ontology based on the terms used in the guidelines issued from W3C/WAI accessibility initiative and the EvalWeb project [10]. Then, we have rewritten these guidelines using the terms in such ontology. The vocabulary contains more than 50 terms related to Web application such as: "*Page*", "*Link*", "*Table*", "*Image*", "*AudioContent*", "*Quotation*", "*NavigationalAid*", "*NavigationalBar*" "*SiteMap*", etc. Figure 3 presents a subset of our ontology concerning the "*navigation mechanisms"* which are used hereafter in our case study.

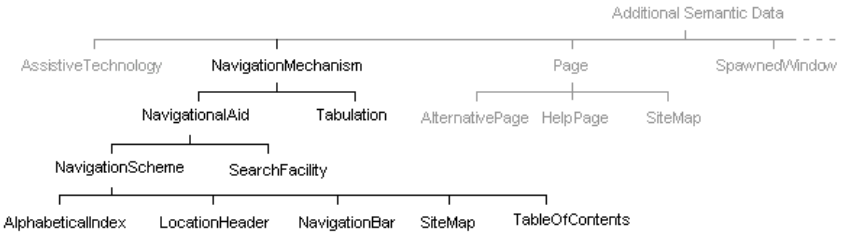

Figure 3 – Part of the ontology describing navigation mechanisms.

This ontology is used for mapping guidelines to artifacts produced during the development process and thus, allowing the automated evaluation during the whole life cycle of Web applications. For example, in a graph-based navigation model, an arc could be interpreted as a "*link*". Since the term "*link*" refers to an element in the common ontology, we can from now apply guidelines concerning links over the navigation model. If a guideline term has no representation in a given model, the rule cannot be verified.

Our guidelines are described using XML files which allow the automatic processing by business rule engines such as Drools[3]. The XML description includes pointers to the corresponding artifacts and libraries used to run out the inspection.

Figure 4 shows an example of guideline (e.g. "*Provide persistent links to the home page and high-level site categories*") translated to the Drools format. Precisely, guidelines described in this way have 3 main parts: a) elements identification, b) test performed to check the guideline, and c) error message if the test fails. The elements identification part contains 3 parameters: "*errorList"* (line 3) which points out to a Java class describing the errors collected during the evaluation; "*page"* (line 6) is the artifact's element assessed in this guideline; and "*domUtils*" (line 9) which refers the library used to run out the inspection over the artifact. In our example, the tests are identified by the <*java:condition*> tag (lines 12 and 13). A

[3] http://drools.org/ (Mars 9th, 2006)

condition should contain method calls the library assigned above. Actually, this library has been implemented by the system expert to handle the ontology and the artifacts. Once this library has been implemented, the rule editor is parameterized to use this library. Finally, when a rule satisfies all the conditions the consequence is thrown. In this example, an error is added to the current list of errors (see line 15).

```
1  <!--[26](A) Provide persistent links to the home page and high-level site categories -->
2  <rule name="[26]">
3    <parameter identifier="errorList">
4       <class>error.ErrorList</class>
5    </parameter>
6    <parameter identifier="page">
7       <class>dom.Page</class>
8    </parameter>
9    <parameter identifier="domUtils">
10      <class>util.DomUtils</class>
11   </parameter>
12   <java:condition> domUtils.isLinkedToMainPage(page) !=true </java:condition>
13   <java:condition> domUtils.isLinkedToImportantPages(page) !=true </java:condition>
14   <java:consequence>
15      errorList.addErgoError("\n\nFailed! [26] Provide persistent links to the
16      home page and high-level site categories [A] \n for State: ", state);
17   </java:consequence>
18 </rule>
```

Figure 4 – Example of guideline descriptions in Drools format.

4. RUNNING OUT AN EXEMPLE

Our purpose is to evaluate guidelines in every phase of the development process. However, it is worth noting that we don't have the same information about the Web site whether we are at the specification level or at the implementation level for instance. For example, in the specification phase we don't have information concerning the Web page contents whereas this information is fully accessible at the implementation phase. So the rules engine processing guidelines must deal with different artifacts at each step. To illustrate such an approach we present hereafter an example of automated verification at each step of development process.

4.1 At the Specification Phase

Design of web user interfaces includes design for structure, navigation and appearance. During the specification several models support the design activity (e.g. organizational model, data model, navigation model, etc.). According to the space available, we just present hereafter the navigation model.

To deal with the navigation issues we employ the SWC notation [16]. SWC is a typical exemplar of a finite state machine. Figure 4 shows a view at glance of the elements of SWC notation.

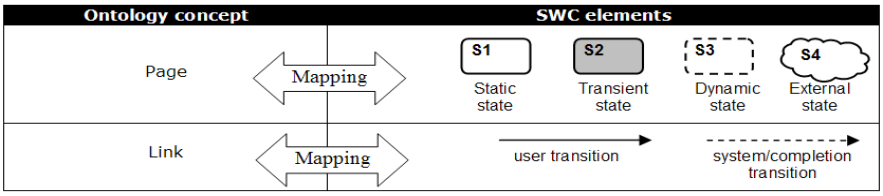

Figure 5 - Navigation modeling of the SPIDER Web project Web site.

In Figure 4, at the top level, a compound state named *"SpiderWeb project Web Site"* group all states taking part of the Web site. Rounded-corner rectangles (i.e., *"main intro"*, *"first workshop"*, *"schedule"* and *"publications"*) are *static states* in the SWC notation which means they refer to pages with static content. External or foreign Web sites (i.e., *"Capes"*, *"Cofecub"*, *II-UFRGS"*, *'LIIHS-IRIT"* and *"IHC'2002"*) are depicted by cloud-like symbols (i.e. *external states* in the SWC notation). The labelled transitions from *"t1"* to *"t13"* correspond to the links between pages. (Refer to [16] for more details about the notation).

As SWC is a navigation model it allows the representation of the concepts such as "page" and "link" by means of the elements "state" and "transition" (see Figure 6). Thus, the state element (static, transient, dynamic or external states) can represent the concept of 'page' in our ontology and vice versa. Likewise, a user, system or completion transition will represent the concept of 'link'.

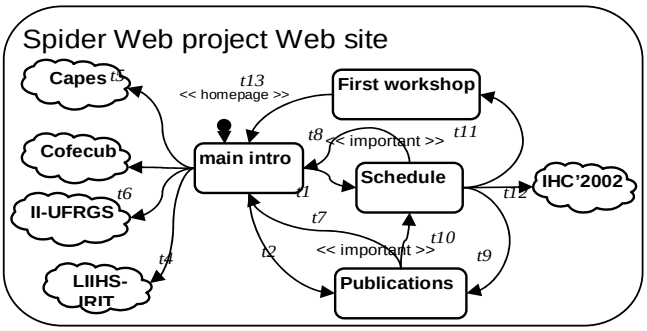

Figure 6 – Mapping between the ontology terms and the SWC elements

After having completed the mapping of SWC elements to the ontology, navigation guidelines "*Each **page** must have a **link** to it*" as "*Each **state** must have a **transition** pointing to it*" can be translated to the following rule: "*Each state must have a transition pointing to it*".

4.2 At the Design Phase

By the end of the design phase sketches providing the layout and CSS style sheets specifying presentation properties are produced. When drawing the sketches, designers can typically point out where the "*navigation bar*" and "*navigation aids*" should be placed on the Web page. However, a "*navigation bar*" can be implemented in many different ways in the HTML code, for example as a set of links, an image map, a Java script, etc. So, we have to find out a way to map the "*navigationBar*" to any possible HTML elements.

Hereafter, we present a solution based on CSS templates to intermediate the mapping. CSS templates help at describing the locations of content and thus creating a flexible mechanism for pointing elements into the HTML since they are often used for presentation issues. Another possibility for this mapping is XLink[4]. Let's consider the description of the element "*Navigation Bar"* in a Web page which refers to the identifier "my*NavigationBar*" located in a CSS file as presented by Figure 7.

```
#myNavigationBar {
   position: absolute;
   top: 4em;
   left: 1em;
   background-image: url(/images/background.gif);
}
```

Figure 7 - Description of a navigation bar in a CSS template.

In that way, we have a particular region on the Web pages called "my*NavigationBar*" on which we apply a particular look & feel. However, "my*NavigationBar*" has no semantics concerning the ontology. Indeed, two or more CSS elements having different look & feel (ex. text-based menu, and image based menu) might have the same semantic (ex. "*NavigationBar*"). In order to give more flexibility to designers, we employ an XML file for mapping CSS elements and the elements of our ontology, as presented in Figure 8.

```
<?xml version='1.0' ?>
<SemanticData xsi:noNamespaceSchemaLocation="semanticData.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
...
   <NavigationalAid  type="NavigationBar"  selector="#myNavigationBar"/>
...
</SemanticData>
```

Figure 8 – CSS description for the identifier "*navigation bar*".

[4] At: http://www.w3.org/XML/Linking (June 27th, 2001)

4.3 At the Implementation Phase

At the implementation level, the content of the Web pages is created by developers. So, the evaluation will now be carried out on the HTML code. The mapping between ontology and HTML tags is done by associating tags and ontology terms in a similar way we have done with SWC and CSS elements. Figure 9 shows how the mapping can be done according some artifacts produced at different steps of development process.

	Specification	Design	Implementation
Ontology concept	**SWC elements**	**CSS elements**	**HTML elements**
Page	State	-	<html>
Link	Transition	a: link	<a>
Navigation Aid	-	#myNavigationbar	<a href=" ..." class="myNavigationbar"

Figure 9 – Ontology mapping for the identifier "*navigation bar*" according several artifacts.

Besides, we can combine DOM elements, CSS templates, XML files describing templates semantic and SWC models to apply guidelines that could not be checked using each of these artifacts alone. So, we can interpret a guideline according to the mappings done in the early phases between the different models and the ontology. For example: "*Each page must contain <a> elements whose targets lead to each page with a 'homepage' or 'important' semantics*'". Like in the previous phase, we manage to distinguish these different pages according to the additional semantic data we add in the XML file. In this step, we can automate evaluation by combination information issued from different artifacts. For example, we can indicate that the "*main intro*" page (see Figure 5) has type "*homepage*" because contrary to the SWC model the DOM model doesn't give information about the type of one page. This mechanism was possible in SWC by means of the stereotype << *homepage* >> (see [16] for details). By combining information HTML files and SWC models it is possible to evaluate the guideline "*Each page must contain a link whose targets lead to each page with'homepage' or 'important' semantics*".

4.4 At the Evaluation (or Deployment) Phase

During the deployment phase the prototype is running so, we should run out the final evaluations before delivering the application to the end users. As mentioned before, the automated evaluation of guidelines does not cover all kind of guidelines. So, at this step, our evaluation tool helps by indicating to the designers the set of guidelines that should be manually inspected and which models and artifacts are concerned by these guidelines. The evaluator is therefore assisted throughout the evaluation with a list of errors/warnings constantly displayed on the screen.

5. TOOL SUPPORT

The edition of guidelines is supported by an editor and a rule engine inspector integrated into the platform Eclipse as presented in Figure 10. This editor supports the edition of rules in natural language by giving access to the vocabulary defined by both the ergonomics expert and the system expert in the early phases. In order to be automated, rules entered in natural language are completed with the corresponding mapping to artifacts.

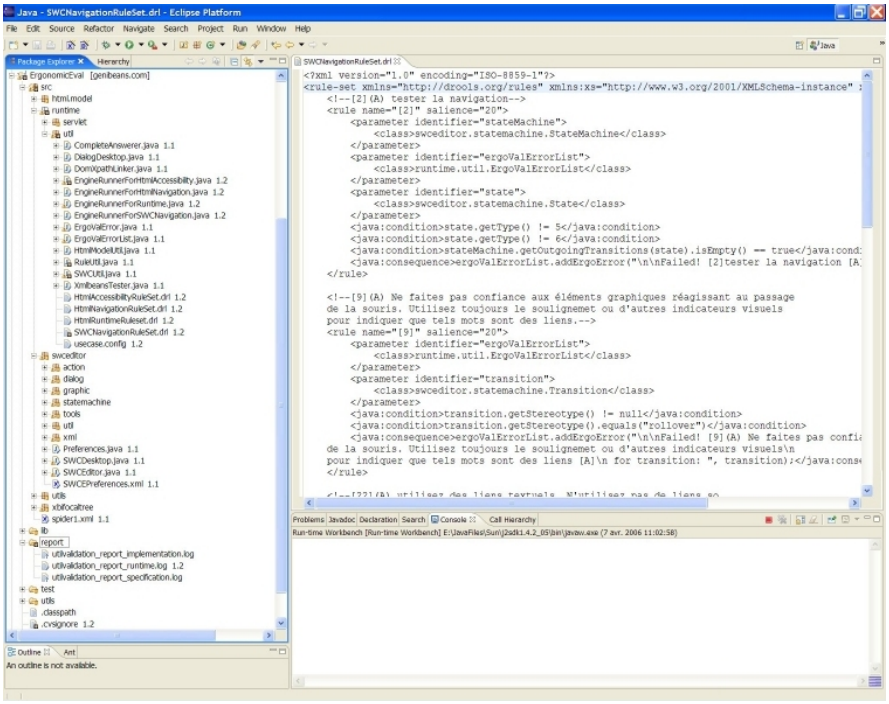

Figure 10 – Tool support integrated into the platform Eclipse.

Each rule is then stored in a knowledge base in a Drools format as presented by

Figure 4. We also have implemented the knowledge base and the rules engine. The rule engine evaluates guidelines contained in the knowledge base throughout the development process. It takes appropriate models as input according to the phase where the rules set is evaluated. As many models are used from the specification phase to the deployment phase many efforts have to be done to address each model. Indeed, if one ergonomic rule is expressed with a unique language in the rule editor, its evaluation is much

more complex: from one model to another one the rule's expression is not the same. After each evaluation a report is generated.

6. DISCUSSION AND FUTURE WORK

Many studies have shown that careful application of guidelines had positive impact on usability [12]. Automated evaluation of guidelines is a way to employ usability methods on Web design.

This work presented a general architecture and a tool support enabling automated inspection of guidelines in very early phases of development. On one hand, our approach is based on the inspection of artifacts available at each phase of the development process. On the other hand, we propose an ontology enabling the mapping of concepts described in the guidelines and the elements found in the artifacts. The description of guidelines should follow a particular ontology, which requires a compromise between the system expert (which knows all artifacts and models used to build Web applications) and an ergonomics expert. But once the common vocabulary has been established, the guidelines can be described in a computational representation allowing the automated inspection.

The implementation of the corresponding mapping between guidelines and artifacts is very costly, as we have experienced, because we have to code a particular library to every artifact used in the inspection. However, once we have established the mapping to a given model (i.e. navigation model SWC) all new project users can benefit from these models, thus reducing the costs over time. At present we have implemented such an approach for automated checking of guidelines over SWC navigation models, CSS and HTML pages, thus covering the phase from specification to deployment of the Web site.

One of the main drawbacks of our approach is that designers should specify all elements of the user interfaces at different levels (model, templates, etc.) and if they miss to clearly identify an element, it will not be checked by our tools. In the future, this task can be alleviated by appropriated authoring environments supporting the design and implementation of Web sites. Future work will include the extension of such an approach for other artifacts produced during the development process. In order to justify the added value of our approach, we intend to compare the results of usability evaluation with our tool with other approaches for usability evaluation.

Even thought the approach for guidelines verification presented here shares a lot of concepts with the Web semantic initiative, none Web semantic technology was used in our implementation. The reason for not

using technologies such as RDF, for example, to describe ontology relays on the low performance of currently available rule engines based on the RDF standard. This pragmatic choice was constrained by our industrial partner which sponsors this project.

We have presented here our preliminary results of a method to improve the use of automated tool for guideline inspection throughout the development process of web applications. Our aim is to get more benefits from a tool support helping to apply in an automated manner ergonomic knowledge on the artefacts produced during the design process. Our ongoing work focus on the description of a decision process model that will generalize our contribution and make it in accordance with actual design process and the applicability of quality models such as discussed in [13].

At the present, we just have employed such as an approach for a few artifacts (i.e. navigation models, templates, CSS and HTML descriptions). Some empirical findings suggest its use on broader set of artifacts, including artifacts related to the structure of the information architecture. We plan to verify this hypothesis in the near future.

ACKNOWLEDGMENTS

This research is partially supported by the WebAudit project, ANRT-CIFRE and Genigraph (http://www.genigraph.fr), Genitech Group.

REFERENCES

[1] Ardito, C.; Lanzilotti, R.; Buono, P.; Piccinno, A. A tool to Support Usability Inspection. In Proceedings of ACM Advanced Visual Interfaces (AVI'2006), Venice, Italy, May 23-26, 2006.

[2] 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. Proceedings of 4th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2002, Kluwer Academics Pub., Dordrecht, 2002, pp. 337-348

[3] Brajnik, G. Automatic web usability evaluation: what needs to be done? In Proceedings of 6th Human Factors and the Web Conference, Austin, Texas, June 2000.

[4] Campos J. C., Harrison M. D. Formally Verifying Interactive Systems: A Review.Eurographics Workshop on Design, Specification andVerification of Interactive Systems (DSV-IS'97), Granada,Spain, 4-6 June 1997. p. 109-124

[5] Farenc, C., Palanque, P., Bastide, R. Embedding Ergonomic Rules as Generic Requirements in the Development Process of Interactive Software. 7th IFIP conference on human-computer interaction (Interact'99), Edinburgh, Scotland, August 30th-September 3rd 1999.

[6] Ivory, M. Y. Automated Web Site Evaluation: Researchers' and Practitioners' Perspectives. Kluwer Academic Publishers. Dordrecht, The Netherlands. 2003. 200p.

[7] Javaux, D., Colard, M.-I., Vanderdonckt, J. Visual Display Design: a Comparison of Two Methodologies, in Proc. of 1st Int. Conf. on Applied Ergonomics ICAE'96 (Istanbul, 21-24 May 1996), A.F. Özok & G. Salvendy (eds.), Istanbul - West Lafayette, 1996, pp. 662-667.

[8] Matera, M.; Costabile, M. F.; Garzotto, F.; Paolini, P. SUE inspection: an effective method for systematic usability evaluation of hypermedia. IEEE Transactions on System, Man and Cybernetics. v. 32 (1), January 2002.

[9] Palanque, P., Farenc, C., and Bastide, R. 1999. Embedding ergonomic rules as generic requirements in the development process of interactive software. In Proceedings of IFIP TC13 Seventh International Conference on Human-Computer Interaction (Edinburgh, Scotland, August 1999). Amsterdam, The Netherlands: IOS Press

[10] Scapin, D. et al. Conception ergonomique d'interfaces web: démarche et outil logiciel de guidage et de support, INRIA Technical Report of EvalWeb project. INRIA-Université de Toulouse 1-UCL. December 1999.

[11] Scapin, Dominique; Leulier, Corinne; Vanderdonckt, Jean; Bastien, Christian; Farenc, Christelle; Palanque, Philippe, and Bastide, Rémi. Towards automated testing of web usability guidelines. 6th Conf. on human factors & the web; Austin, Texas, USA. 2000.

[12] Scholtz, J., Laskowski, Sh. Developing Usability Tools and Techniques for Designing and Testing Web Sites. In Proceedings of the 4th Conference on Human Factors and the Web. Basking Ridge, USA, June 1998.

[13] Seffah, A.; Gulliksen, J.; Desmarais, M. (eds.) Human-Centered Software Engineering: Integrating Usability in the Software Development LifeCycle. Springer, Dordretht, The Netherlands, 2005. 391 p.

[14] Vanderdonckt, J.; Farenc, C. Tools for Working with Guidelines: Annual Meeting of the Special Interest Group (TFWWG'2000). Springer publishers, London, UK. 2001.

[15] Vanderdonckt, J. Development milestones towards a tool for working with guidelines. Interacting with Computers 12(2): 81-118 (1999).

[16] Winckler, M., Palanque, P. StateWebCharts: a Formal Description Technique Dedicated to Navigation Modelling of Web Applications. International Workshop on Design, Specification and Verification of Interactive Systems - DSVIS'2003, Funchal, Portugal, June 2003. (Lecture Notes)

[17] Winckler, M., Farenc, C., Palanque, P. Automatic Evaluation for the Web: How Improve Navigation Guidelines. Workshop on Automated Testing in ACM Conference on Computer-Human Interaction – CHI'2002, Minneapolis, USA, April 21-22, 2002.

[18] Xiong, J., Farenc, C., Winckler, M. Vérification de règles ergonomiques sur un modèle de navigation des applications Web. In proceedings of IHM'2004, Namur, Belgium, August 30th-September 3rd 2004. ACM, p. 259-262.