DOCTOR OF SCIENCES

Methodology for automating web usability and accessibility evaluation by guideline

Beirekdar, Abdo

Award date:
2004

Awarding institution:
University of Namur

Facultés Universitaires
Notre-Dame de la Paix

A Methodology for Automating Guideline Review of Web Sites

Abdo Beirekdar

Thesis submitted in fulfillment of the requirements for the degree of Doctor of Sciences
(Computer Science Option)

- August 30th, 2004 -

Director:	Professor M. Noirhomme-Fraiture
Co-director:	Professor J. Vanderdonckt, Université Catholique de Louvain, Belgium
Jury:	Professor F. Bodart
	Professor J.-L. Hainaut (President)
	Professor Ch. Kolski, Université de Valenciennes, France
	Professor Ph. Palanque, Université Paul Sabatier - Toulouse III, France

Institut d'Informatique
NAMUR

Chapter 6

Tool-Suite for Automated Web U&A Evaluation

6.1 Introduction

In this chapter we will focus our attention on the tools we are developing as a support for the proposed methodology. We start in section 6.2 by precising the requirements of them and describing their global architecture. In section 6.3 we describe a possible implementation of this architecture. The development of these tools is not finished yet; we will identify what is done and what is not.

The integrated set of tools is baptized KWARESMI[1] (**K**nowledge-based **W**eb **A**utomated **E**valuation with **RE**configurable guideline**S** opti**MI**zation)].

6.2 Requirements and Architecture

KWARESMI is supposed to meet the following requirements [Beirekdar et al. 2002a, Beirekdar et al. 2003, Smith and Howes 2001]:

- **Be knowledge-based**: it exploits the ergonomic knowledge contained in ergonomic guidelines and re-expresses it in terms of HTML knowledge contained in the semantic of HTML elements.
- **Be Web-oriented**: it works on HTML code of Web pages as their HTML code is accessible, which is not always the case for other user interfaces.
- **Be automated**: it should release evaluators from as many evaluation tasks as possible whether theses procedures can be automated.
- **Be reconfigurable**: to address shortcomings of existing tools, the tool enables evaluators to control both the guidelines inclusion and the evaluation process.
- **Enable guidelines evaluation efficiency**: it should identify potential common ergonomic information among different guidelines. This information should be used to improve evaluation of these guidelines.
- **Support the GDL formal language**: the development of these tools was triggered by the need for a support for the guideline structuring language that we proposed.

By meeting these requirements, the KWARESMI would overcome the major shortcomings of existing automated UAE tools. It is worth noting that

[1] Historical note: Al-Kwaresmi was an outstanding Arabian mathematician who lived around the tenth century. He translated several Indian works on algebra and introduced many new concepts in mathematics. Among these concepts is the “algorithm”, which is pronounced Kwaresmia in Arabic.

KWARESMI is intended to enable the evaluation of any ergonomic guideline (for usability, for accessibility, for WebTV, etc.) as soon as we can find HTML elements that enable the evaluation of this guideline partially or totally. Therefore, the evaluation should not be restricted to specific types of guidelines.

Figure 6.1 depicts the task model of the tasks and sub-tasks that can be accomplished by the tools to evaluate a guideline. We identify the following tasks:

- **Prepare the guideline**: this task concerns the structure of the guideline. We have one of two tasks to accomplish:
    - *Structure guideline*: if the guideline is not already structured, we need to structure it in a GDL-compliant form.
    - *Load the structure*: if this structure already exists.
- **Prepare the evaluation**: this task concerns the configuration of the evaluation session. We have two optional tasks to accomplish. These tasks are optional and a default configuration will be used if the evaluator skips them:
    - *Select evaluation sets*: as evaluation sets correspond to specific aspects of the guideline, we can select the sets to be considered in the parsing and evaluation phases. It is worth recalling that parsing the Web page is based on the evaluation sets: the parser captures usability data related to these sets only.
    - *Configure parsing and evaluation phases*: a tool based on our approach should enable the evaluator to have high control level of the parsing and evaluation phases. This sub-task concerns the selection of some configuration parameters provided by the tool.
- **Conduct evaluation**: after doing all the needed preparations, the evaluator can start the evaluation of a Web page. To do so, he needs to accomplish the following sub-tasks:
    - *Locate the Web page*: we can evaluate local HTML pages or online Web pages. In the first case, the evaluator needs to browse the local hard disk to find the page. In the second case, the evaluator needs to provide the URL of the page and the tool will crawl it (download it locally).
    - *Parse the page*: the end of locating the page triggers the GDL parser that will use the evaluation sets and the parsing parameters to scan the page and capture usability data. When it ends, it calls the evaluation module (GDL evaluator).
    - *Evaluate the page*: the evaluator checks the captured data using the evaluation conditions and the evaluation parameters and generates an evaluation report. The end of this sub-task triggers the report editor.
    - *Visualize the evaluation report*: the generated evaluation report can be consulted under different details levels of the report (see only errors, see passed tests, see statistics about the errors, etc.).

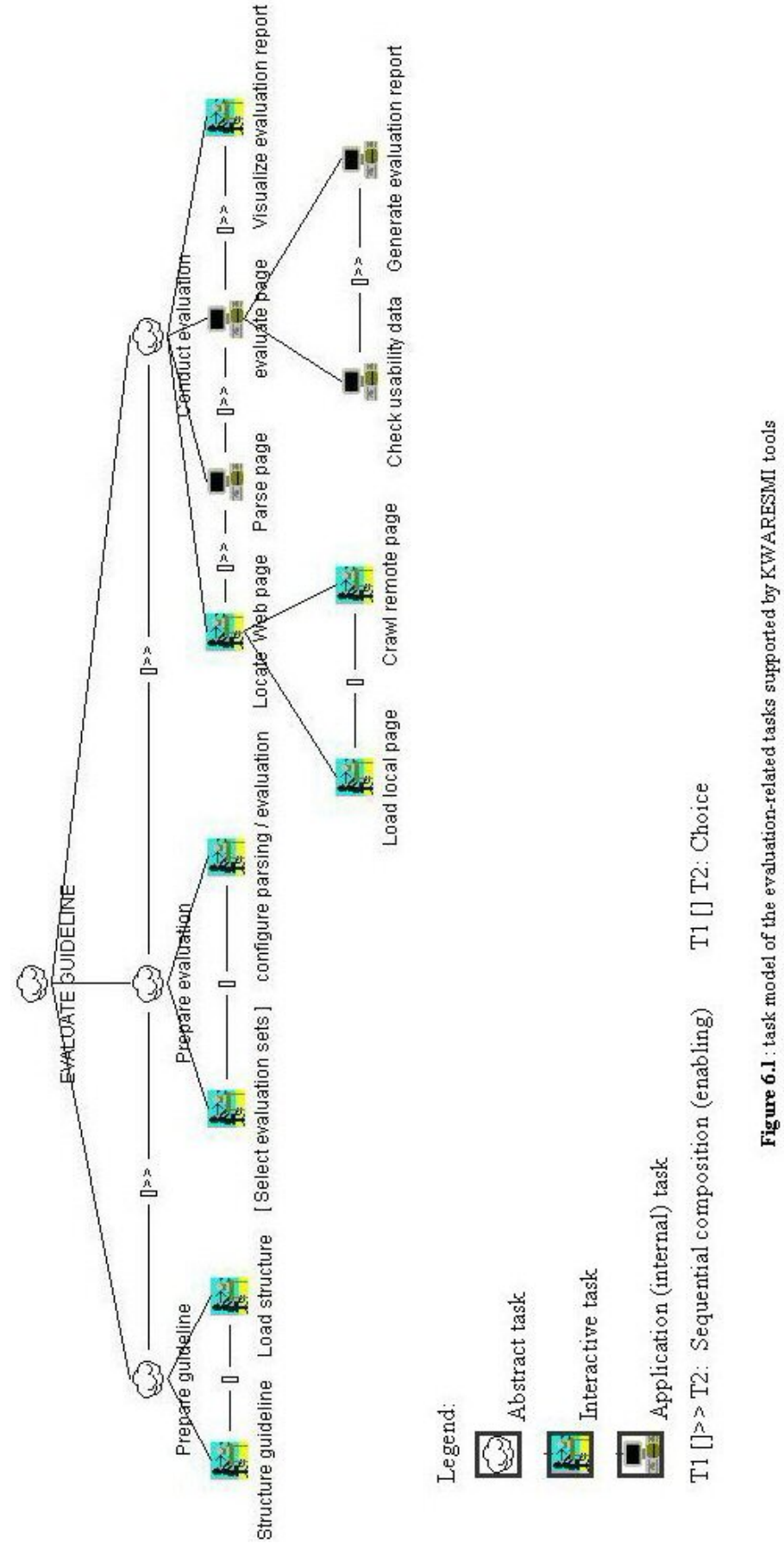

Figure 6.1 : task model of the evaluation-related tasks supported by KWARESMI tools

Figure 6.2 depicts the global architecture of KWARESMI. The proposed tools are aimed at supporting the tasks of figure 6.1.

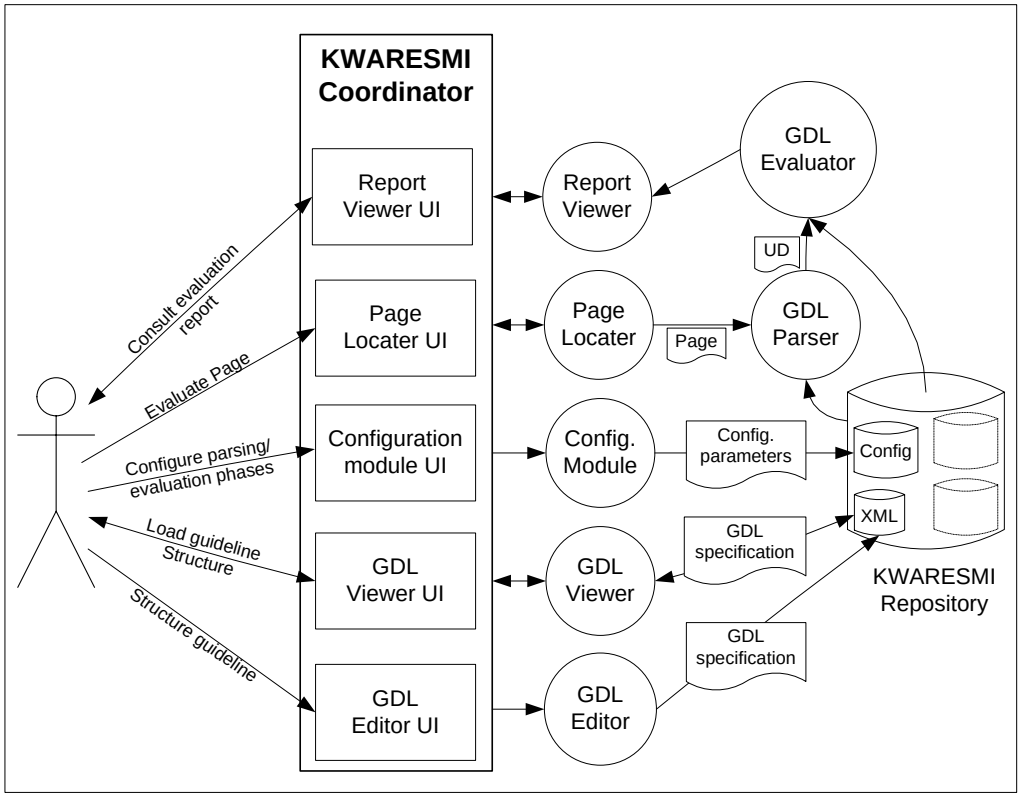

Figure 6.2: Global architecture of KWARESMI

As figure 6.2 shows, a central module plays the role of coordination between the front end components (the GUI) and the back end components of KWARESMI. This modeling is very close to the MVC model and thus, it enables a clear separation between the acquisition of data and its treatment. In addition, it facilitates the reuse of individual modules separately.

The coordinator (obliges) the user to follow the steps of the underlying methodology by presenting a sequence of input dialog boxes corresponding to these steps. The **GDL editor** allows the human factor expert to introduce information about the original guideline, its potential interpretations (at least the default one), and the formal structure of the guideline or the interpretations. **GDL viewer** allows the evaluator to load existing structures. The **configuration module** enables the evaluator to parameterize the parsing and evaluation phases of the evaluated Web pages. After providing all the needed information, the evaluator can start evaluating Web pages by locating them locally or remotely via the **page locater**. The evaluation starts by scanning the page by the **GDL parser** who uses the parsing parameters and the GDL structure of reviewed guidelines to scan the page and capture related usability data. This data is then sent to the **GDL evaluator** who, according to the evaluation parameters and evaluation conditions of the GDL structures, checks the data and generates an evaluation report and delivers it to the report viewer. The **report viewer** enables the evaluator to see the

report content under many views (sort errors, show errors and/or passed tests, show evaluation statistics, etc.).

6.3 Implementation and Application

The current prototype of KWARESMI is implemented in Java Swing (jdk1.5). The repository is not implemented yet. At the moment, structures are introduced manually via a text editor and are saved as standalone XML files.

Following is the presentation of the different KWARESMI modules. We will present them with direct application of implemented ones on some guidelines of chapter 6.

The choice of Java is mainly motivated by the following:

- Java is now widely used for different kinds of applications, and sophisticated Java (free and commercial) IDEs are widely available.
- Java provides high support for XML: parser, generator, convert XML-HTML, etc.
- Java is naturally portable which enables us to deploy the same code on different platforms.
- Java is the best candidate to develop a next-step online version of KWARESMI without big modification of the local version.

As for the KWARESMI repository, MySQL seems to be the best candidate:

- MySQL is a full-featured relational database management system.
- MySQL is portable: it has been ported to almost every platform. This means that you don't have to change your main platform to take advantage of MySQL. And if you do want to switch, there is probably a MySQL port for your new platform.
- MySQL also has many different application programming interfaces (APIs). They include APIs for Perl, TCL, Python, C/C++, Java (JDBC), and ODBC.
- The MySQL database is available at no cost under the GPL open source license.
- MySQL is a very robust database server. It has advanced security measures. It also provides high speed and flexibility levels.

Next, we are going to present the different modules presented in figure 6.2. We will give screen shots of the implemented ones, and present possible designs for non implemented ones.

6.3.1 GDL editor (Not implemented)

We started implementing this module with a database support, but we did not continue because the introduction of evaluation conditions seemed very complicated. We preferred to introduce the specification manually in order to focus our attention on the parsing and evaluation phases. Figure 6.3 and 6.4 show the implemented windows corresponding to the introduction of the guideline and its interpretations, and the specification of evaluation sets. Figure 6.5 depicts a

possible window for conditions specification. These windows have some incompatibility with the latest version of the GDL.

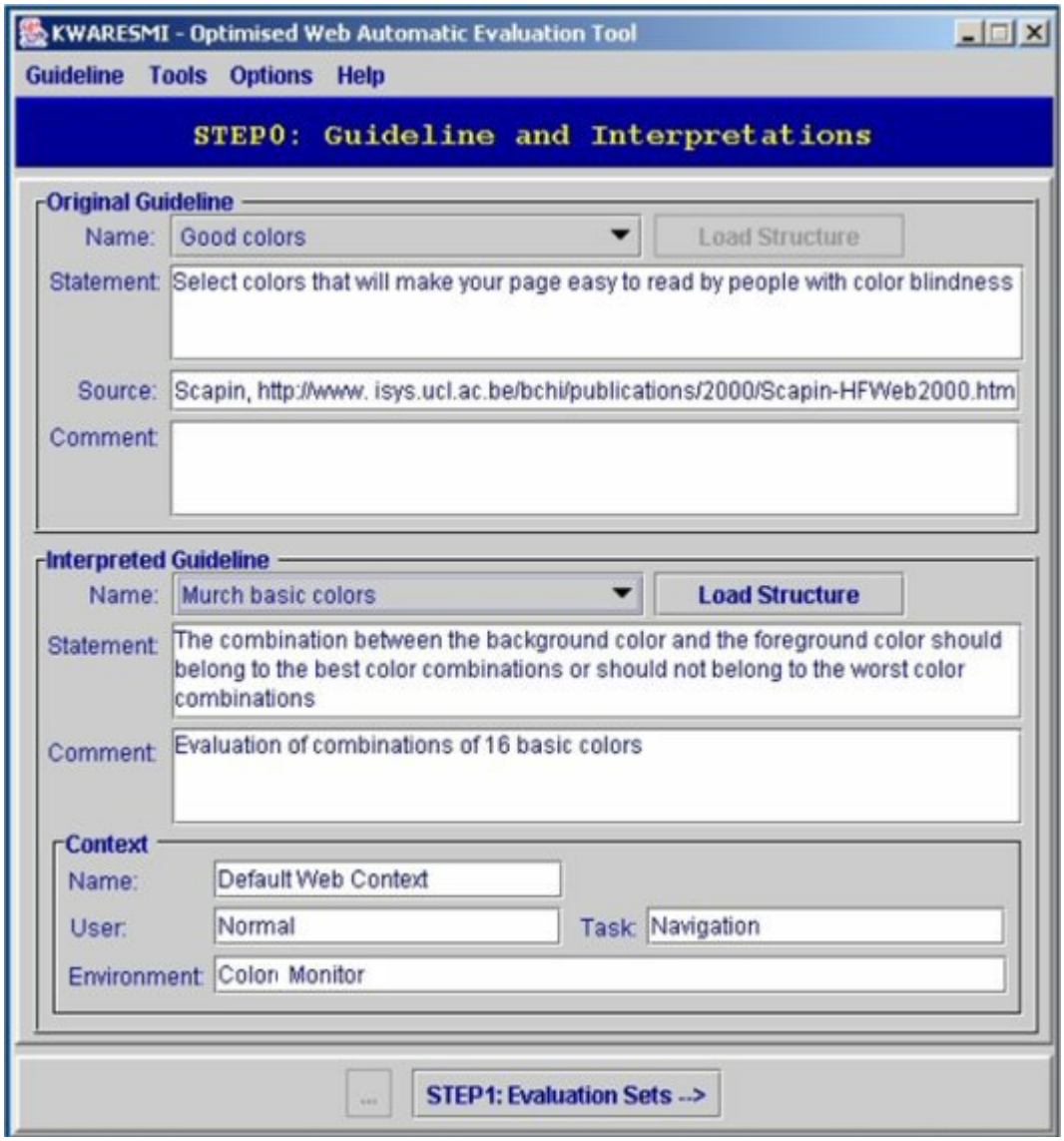

Figure 6.3: Structuring phase -Window1 (will be redesigned): introduce information about the guideline, the interpretation of this guideline if needed, and about the interpretation context. If a formal structure of the original guideline or of its interpretation is already stored in the database, we can load it and pass directly to the next phases.

After providing guidelines information, we specify the formal guideline associated to every interpretation. The formal guideline is composed of two major parts: elements related to structuring information (Figure 6.4) and elements related to evaluation logic (Window 6.5).

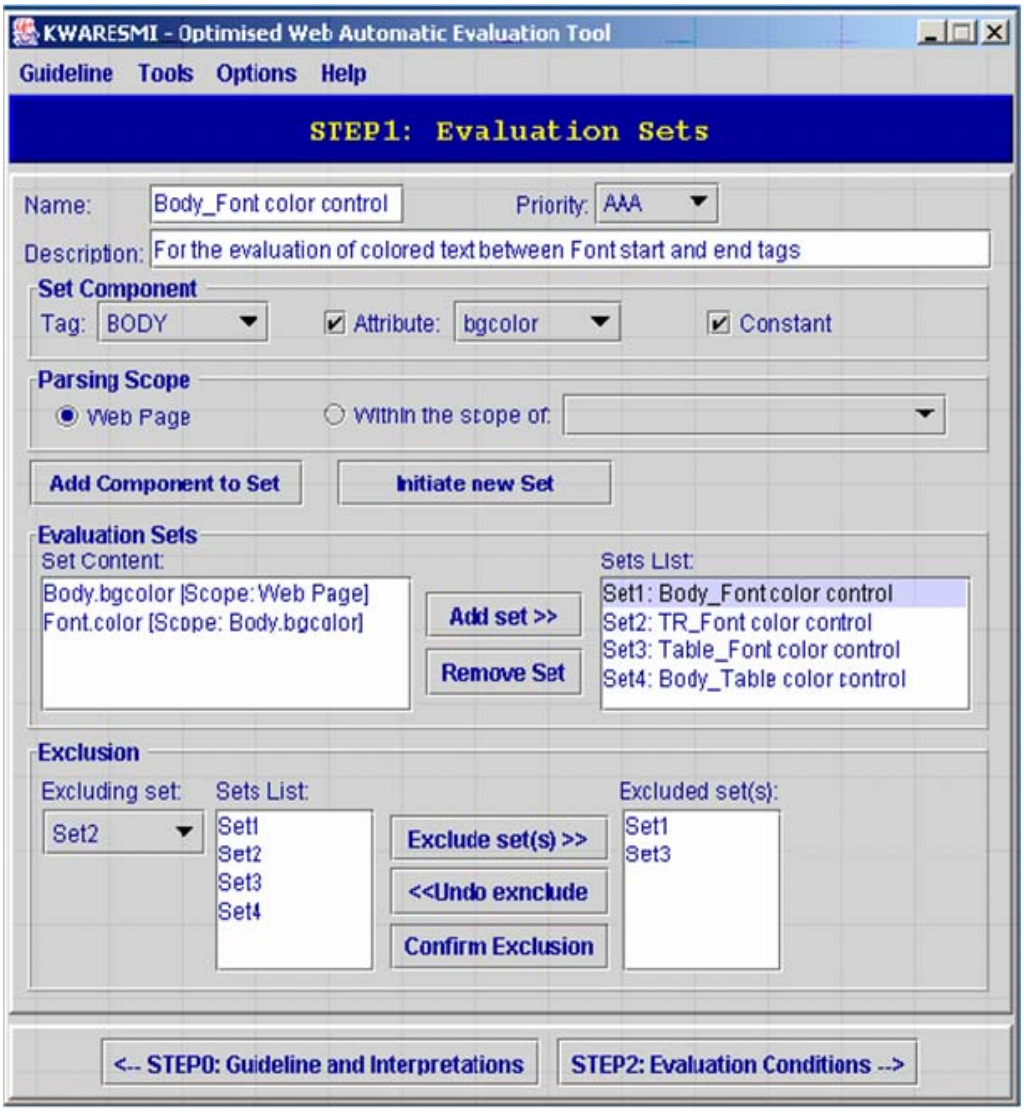

Figure 6.4: Structuring phase - Window2 (will be redesigned): evaluation sets. The editor provides some help by providing the user with the list of all available HTML tags and attributes. He constructs the evaluation sets from HTML elements. During this construction, he provides a default priority level for the evaluation set and the scope of every set element (whole evaluated page or within an element of the same evaluation set). He can also provide a set name and short description to clarify the purpose of the set. Every set is assigned an identifier generated automatically by the tool. After specifying the evaluation sets, the user can specify exclusion relationships among them.

When all the evaluation sets are defined, the user can go to next step to specify the evaluation logic that consists of evaluation conditions (Meta, mapped and direct ones).

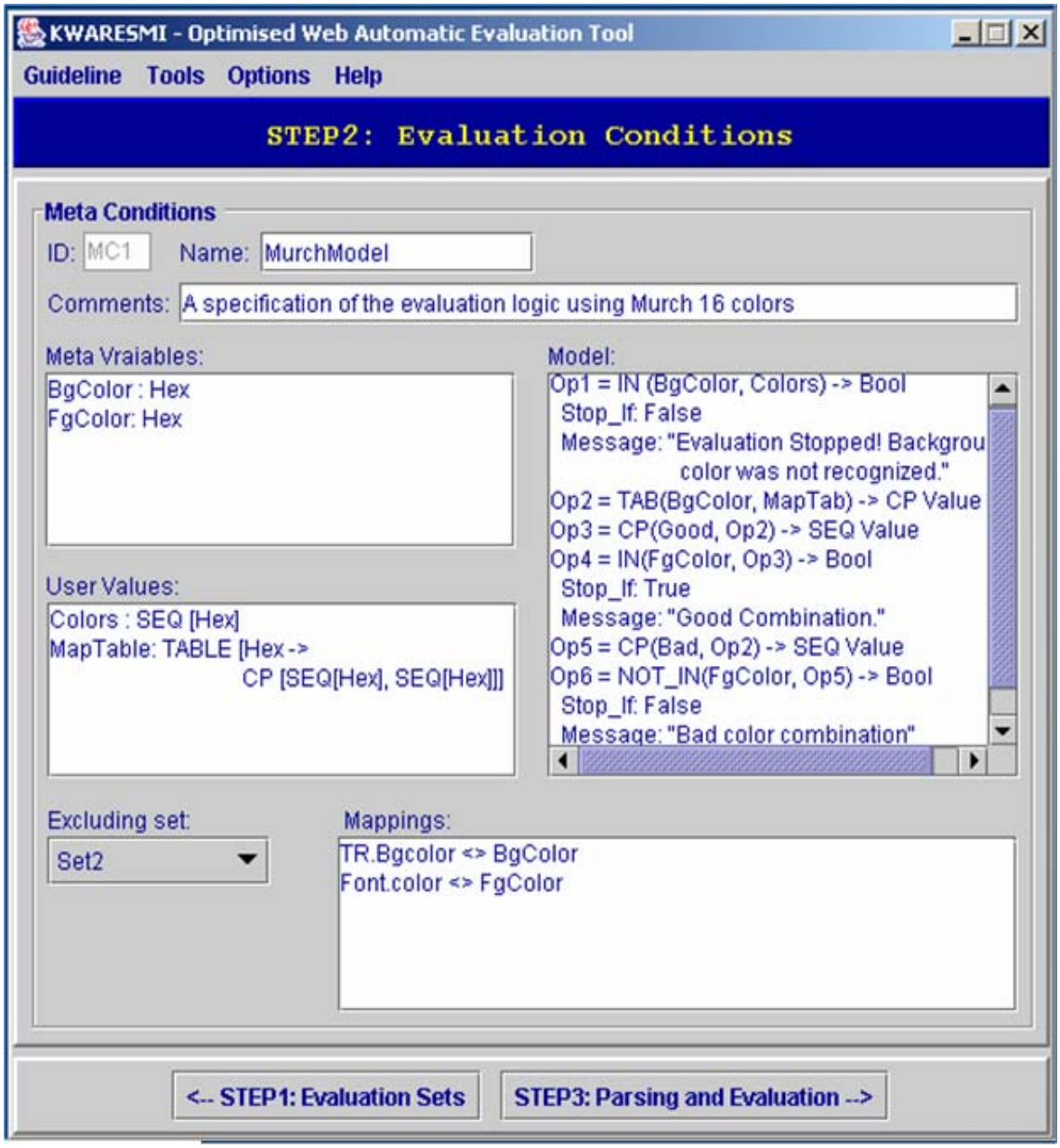

Figure 6.5: Structuring phase - Window3 (Draft, not implemented): evaluation conditions.

6.3.2 GDL Viewer (Implemented)

This module enables the user to load and visualize guidelines structures. As structures are currently saved as separate XML files, the viewer uses the java XML parser SAX and some auxiliary classes of the packages javax.xml.parsers, org.xml.sax, and org.xml.sax.helpers to parse the XML file and extract the structure. The structure is then displayed in the view window (Figure 6.6). The viewer enables the evaluator to select and unselect guidelines or interpretations, to see the evaluation sets of an interpretation, and to select/unselect evaluation sets.

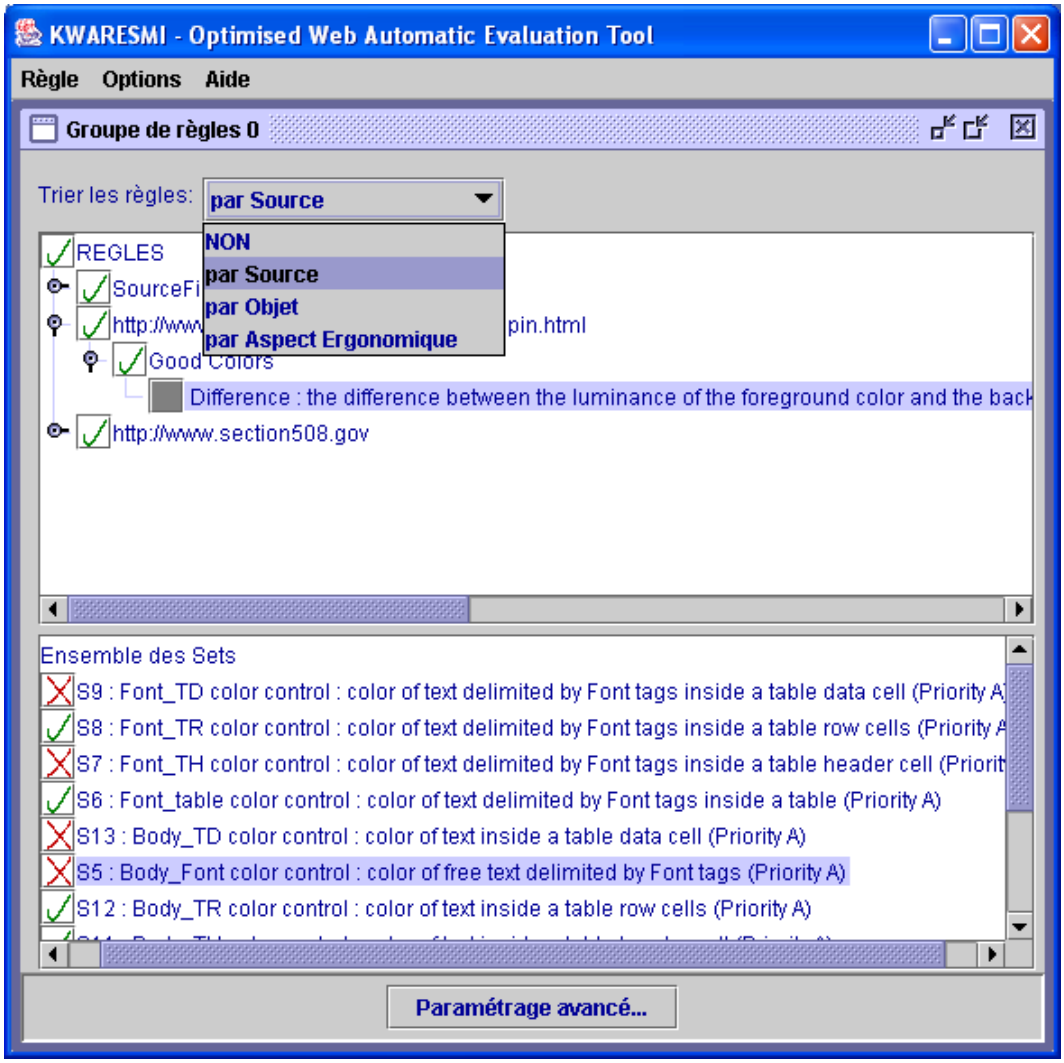

Figure 6.6: GDL Viewer. The evaluator can sort guidelines according to their source, to related ergonomic aspects (information provided in the guideline definition), and by the objects targeted by the guideline. When selecting an interpretation, the composing evaluation sets are displayed. The evaluator can select to evaluate a whole interpretation of some of its evaluation sets. Advanced parameters enable the evaluator to define selection parameters for all the guidelines in the same time: selection by priority (1,2,3) and by objects.

6.3.3 Configuration module (Not implemented)

This module enables the evaluator to configure the parsing and evaluation phases (Figure 6.7).

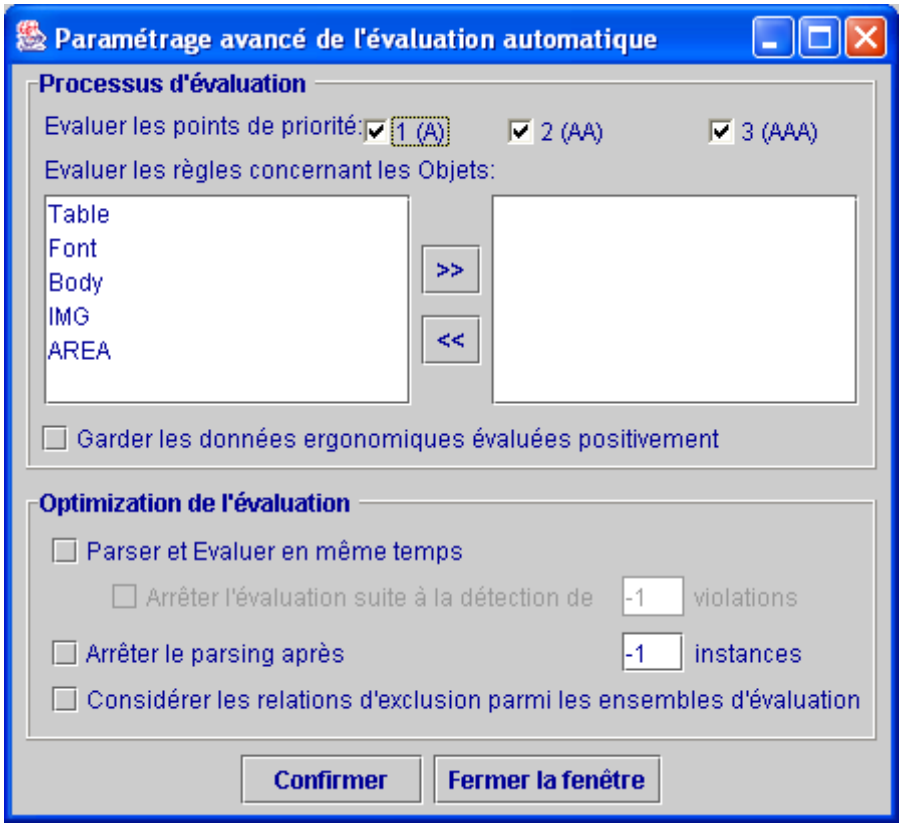

Figure 6.7: Configuration module.

6.3.4 Page locater (Implemented)

This is a very simple module (Figure 6.8). To locate the Web page on the hard disk it calls the predefined Java file browser dialog box. To locate a remote Web page we just need to introduce its correct URL in an edit box. In the later case, the locater calls an auxiliary module to download the Web page and save it in a temporary local file. That task of the locater ends by calling the GDL parser to start its task.

Figure 6.8: the page locater.

6.3.5 The GDL parser (Implemented)

This (invisible) module is currently able to parse a Web page to capture needed data for the evaluation of selected guidelines, but there is no support for evaluation improvements offered by the GDL. The parser needs to do a single

forward scan of the Web page, which is an advantage of our tool over many existing tools that need generally to scan of the page more that once.

6.3.6 The GDL evaluator (Implemented)

This is the core module of KWARESMI. It is currently able to evaluate many guidelines in the same time but on a single Web page. As for the parser, the current evaluator has no support for improvement possibilities offered by the GDL. It examines all the instances of an evaluation set and applies the associated evaluation condition of each of them. It generates an evaluation reports and delivers it to the report viewer to present it to the evaluator.

6.3.7 The report viewer (Implemented)

This module allows the visualization of the evaluation report by the evaluator (Figure 6.12a). The evaluation report contains also some statistics about the evaluation, particularly the evaluation results (Rp, Rn), the number of passed tests and the non-passed tests (Figure 6.12b).

6.4 Case Study

Here we use the current KWARESMI version to evaluate the guideline: "Provide equivalent alternatives to auditory and visual content" [WAI 1999] on the page www.info.fundp.ac.be (Figure 6.9).

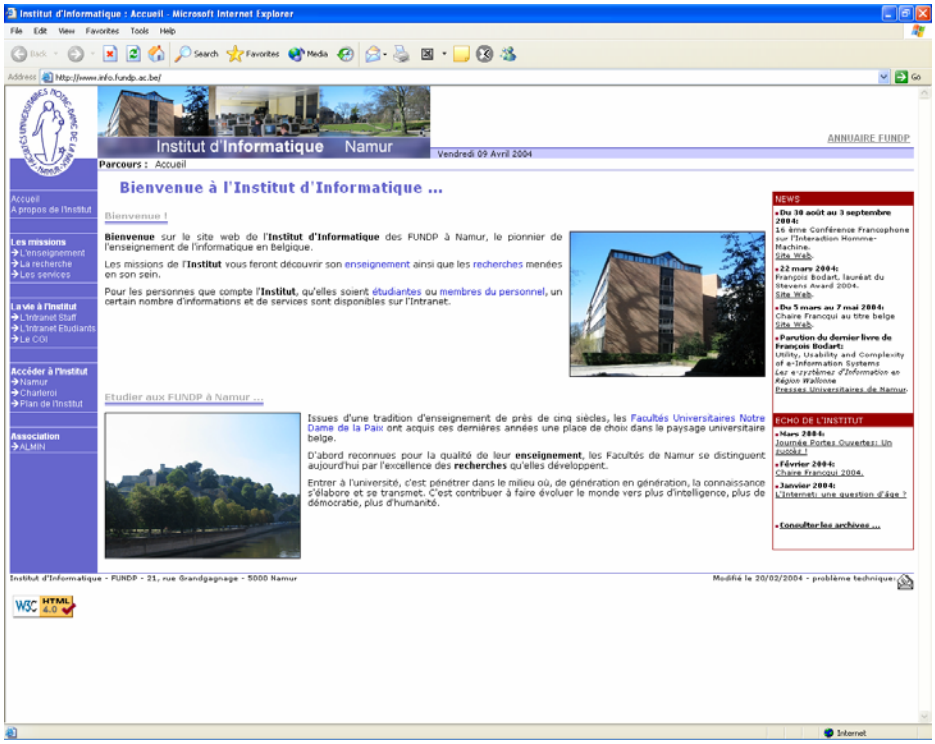

Figure 6.9: www.info.fundp.ac.be

6.4.1 Guideline Structuring

The first thing to do is to specify the GDL structure of the guideline. As the GDL editor is not implemented yet, we use any other editor (text or XML) to

accomplish this task (figure 6.10). the detailed specification process is given in annex B.

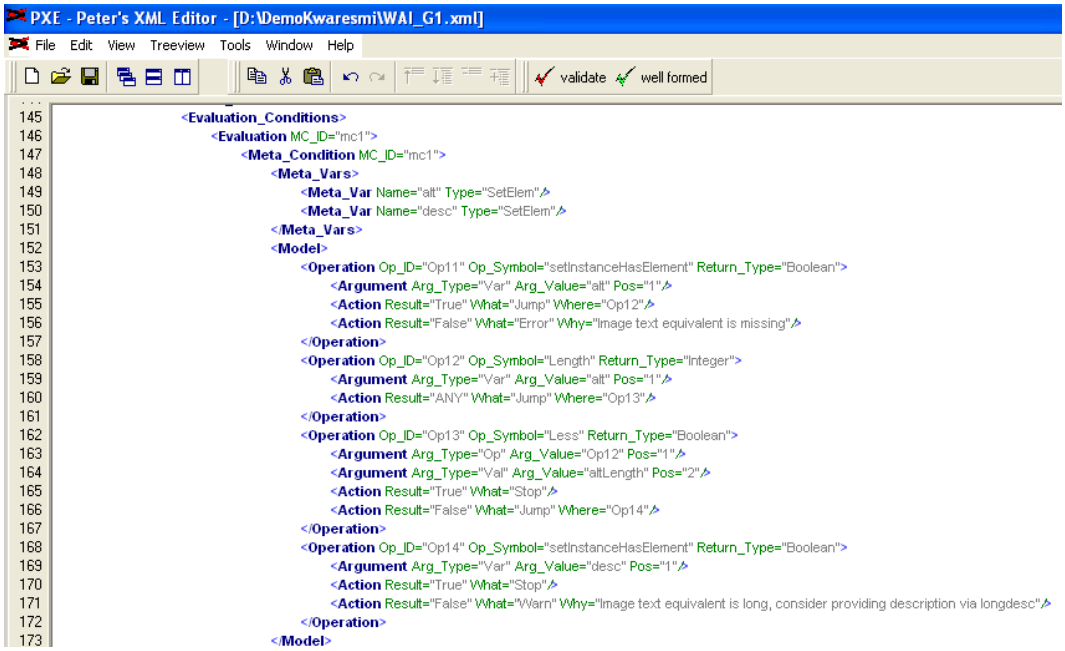

Figure 6.10: a freeware XML editor (©Peter Reynolds, available at www.iol.ie/~pxe)

Now that we have the structure, we can start the KWARESMI viewer to do many things:

6.4.2 Structure visualization

We use the viewer to localize and load the structure. We can then discover its content (interpretation and evaluation sets) and start configuration be deselecting non interesting ones (Figure 6.11).

As advanced parameters are not implemented yet, we pass directly to the evaluation phase.

6.4.3 Page evaluation

We use the page locator to provide the URL of the targeted page: http://www.info.fundp.ac.be, and to call the evaluation module. This module:

- Scans the page,
- Captures data related to the selected evaluation sets,
- Applies evaluation logic on them, and
- Generates the evaluation report.

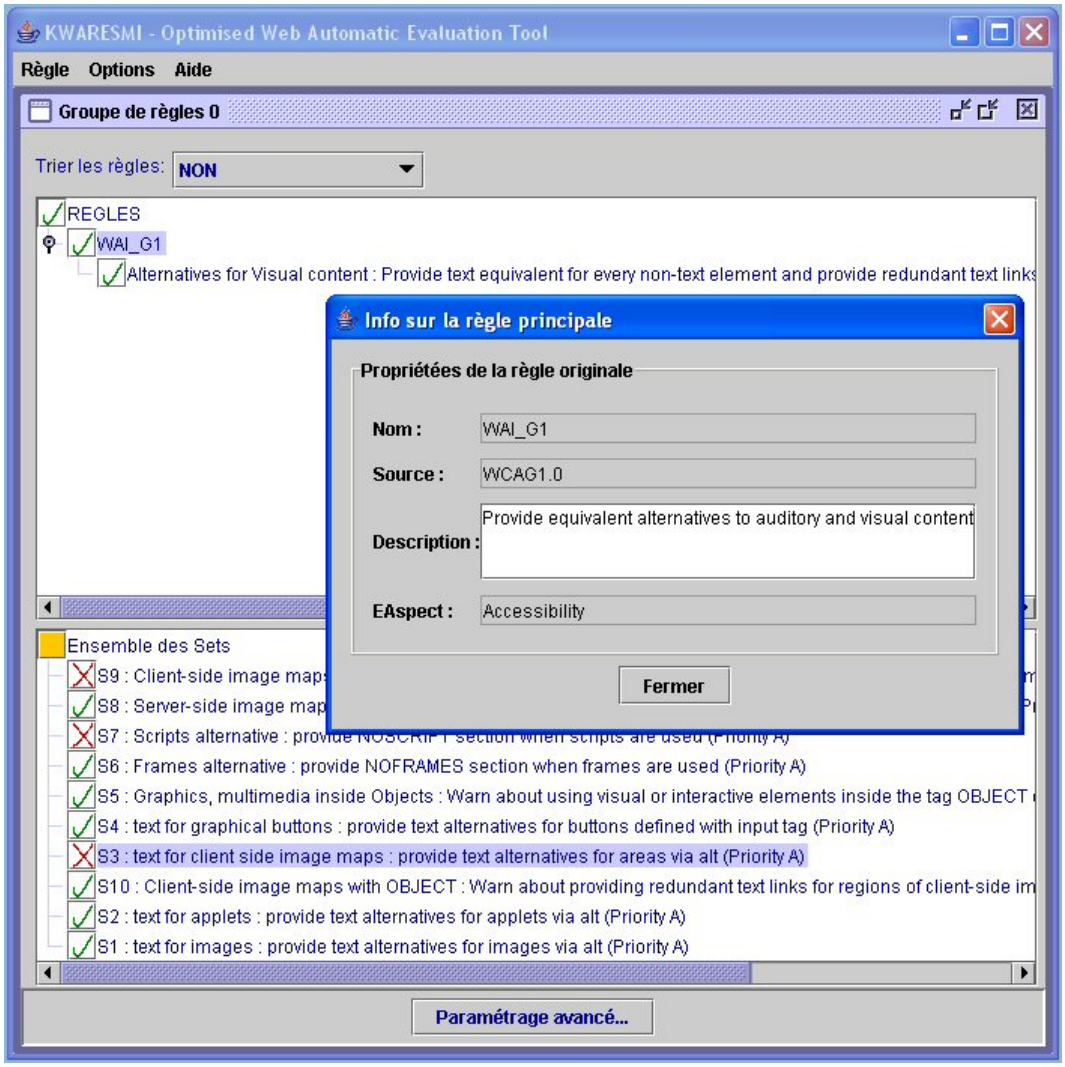

Figure 6.11: Visualization of structure's content (interpretation and evaluation sets)

The evaluation report in our case is shown in figure 6.12a,b.

Figure 6.12a: Presentation of an evaluation report by the report viewer. The evaluator can save the report in a text file, and he can visualize a saved evaluation report. The evaluator can also see the operations (as specified in the formal structure) that the evaluation module executed to generate the result. This is useful when testing the provided structure on some page examples to verify the correctness of the evaluation logic.

Summary

This chapter presented a prototype of the KWARESMI evaluation system. By examining the requirements at section 6.2 and the presented modules, we can say that the current version meets many of them partially or fully, and proves the ability of a complete version to meet them.

- **Be knowledge-based**: the guideline structure makes the link between the natural ergonomic knowledge and formal structure mainly based on manipulating HTML elements to re-express the semantics of the guideline.
- **Be Web-oriented**: all the usability data is captured from HTML code only.
- **Be automated**: many of the evaluation related tasks are already highly automated: examination of the HTML code (parsing), analysis of the code

(evaluation), writing of evaluation report (generated automatically) and computation of evaluation statistics.

- **Be reconfigurable**: GDL viewer, configuration module, and report viewer show a lot of configuration possibilities at many levels.
- **Enable guidelines evaluation improvement**: not proved yet but potentially feasible.
- **Support the GDL formal language**: almost all the modules are directed by information from GDL-compliant guidelines structures.

A lot of work is still needed to obtain a stable and fully functional version of KWARESMI, but the first results are promising.

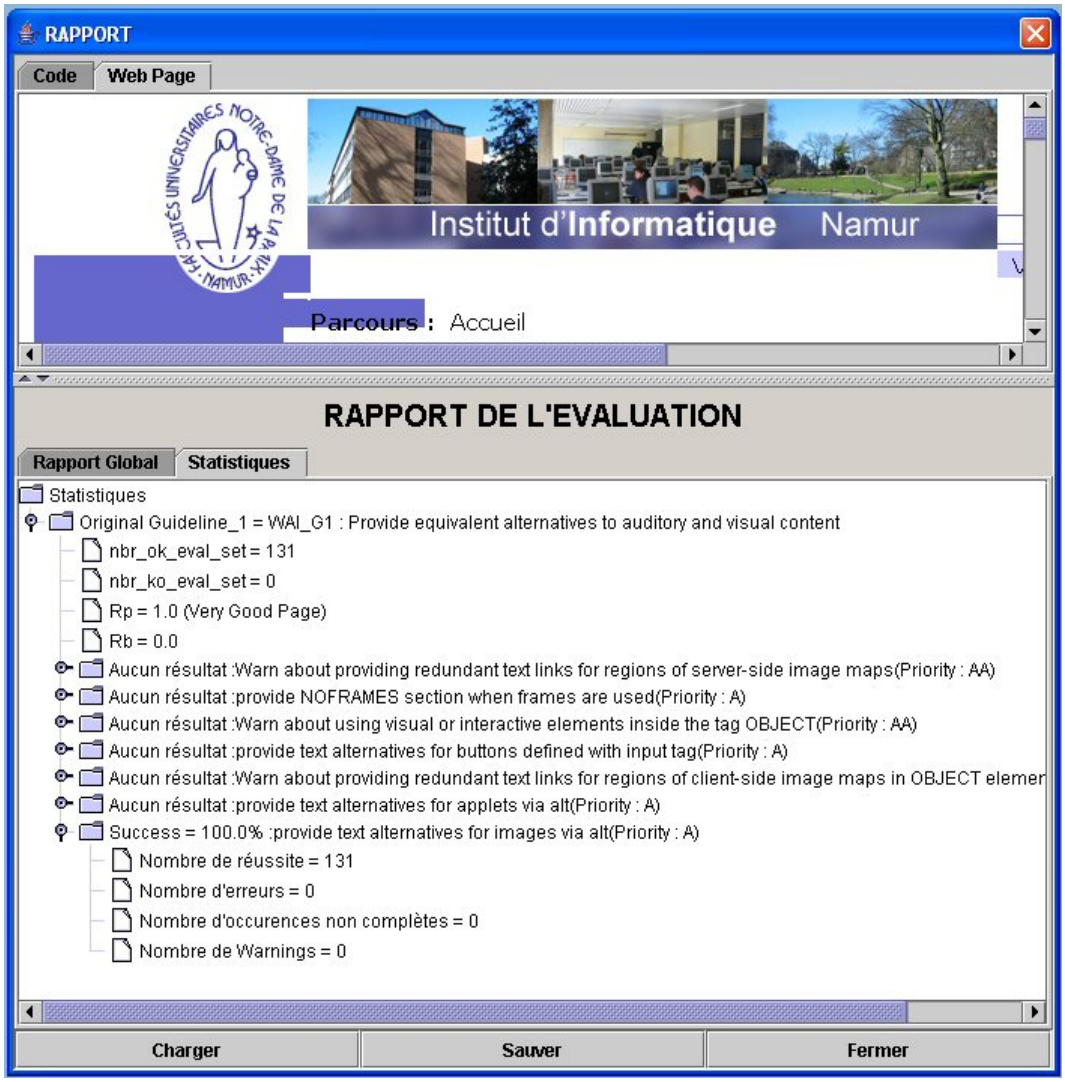

Figure 6.12b: statistical information about the evaluated Web page. We can see that we detected 131 instances of simple images, and that all of them are ok. The final result shows the positive rating of the page according to a simple quality model: positive Rp=(Nbr of OK instances/Total of instances), and negative Rn=(Nbr of KO instances/Total of instances)