TO APPEAR: ICCHP2006

Test Case Management Tools for Accessibility Testing

Sandor Herramhof[1], Helen Petrie[2], Christophe Strobbe[3], Evangelos Vlachogiannis[4], Kurt Weimann[5], Gerhard Weber[5], and Carlos A. Velasco[6]

[1] University of Linz, Institut Integriert Studieren,
Altenbergerstr. 69, 4040 Linz, Austria,
sandor.herramhof@jku.at
[2] University of York, Department of Computer Science,
Heslington, York, YO10 5DD, UK,
petrie@cs.york.ac.uk
[3] Katholieke Universiteit Leuven, Dept. Electrical Engineering - SCD,
Kasteelpark Arenberg 10, 3001 Heverlee, Belgium,
christophe.strobbe@esat.kuleuven.be
[4] University of the Aegean, Department of Product and Systems Design,
Ermoupolis, Syros, GR83200, Greece,
evlach@aegean.gr
[5] University of Kiel - Multimedia Campus Kiel,
Theodor Heuss Ring 140, 24143 Kiel, Germany,
{k.weimann, g.weber}@mmc-kiel.com
[6] Fraunhofer-Institut fuer Angewandte Informationstechnik FIT,
Schloss Birlinghoven, D53757 Sankt Augustin, Germany,
carlos.velasco@fit.fraunhofer.de

Abstract. Two tools are presented which support test case management for accessibility test suites. Creating test suites for the Web Content Accessibility Guidelines 2.0 is one major objective of the EU-funded project BenToWeb[1]. Parsifal is a desktop application which easily allows editing test description files. Test description files compose an XML layer containing descriptive information about the particular test cases. Amfortas is a web application which allows controlled evaluation of the test suites by users. Controlled in that sense means, that Amfortas not only stores the evaluation results, but also is aware of the physical and technical condition of the evaluator.

1 Introduction

According to the AMFORTAS methodology [**?**] a test case consists of test material (e.g. an HTML file) and a test case description. The Test Case Description Language TCDL [**?**] developed within the scope of BenToWeb, not only states whether a test case passes or fails. For user evaluations it additionally provides guidance on the combination of assistive technologies/user agents/devices a test

[1] http://www.bentoweb.org

case should be tested with. The creation of test cases (a complete set matching each of the requirements of a high-level specification like WCAG) is costly and work-intensive, but it has the advantage that false negatives are much easier to find than in the evaluation reports of real websites [**?**].

Preparation, editing and management of user, expert or manual tests can be improved if appropriate software tools are available. Two tools have been developed to support the editing process of the test case description files and to manage the test case evaluation procedure.

2 Parsifal - Test Case Editor

Parsifal is a graphical Test Case editor for editing user test descriptions as defined by TCDL1.1 specification [**?**]. Test case description files are written in XML. Since work with XML documents in text editors is not very comfortable and error-prone a graphical editor was implemented to ease editing XML test case description files. Parsifal is implemented in C# using .Net Framework 1.1. A setup routine for easy installation and runtime configuration is provided for Parsifal. In Parsifal parallel work on test case descriptions is guaranteed by using a Concurrent Versions System (CVS), for managing multiple submits from different partners at a time. The Annotation facility of CVS is used to comment changes made on test cases.

Parsifal consists of three main components. A graphical user interface component, which describes the graphical editor. A CVS component, which deals with CVS file synchronization between editor and CVS server. And a serializer component, which is responsible for reading, writing, previewing and printing test case documents. Standard requirements for editing, saving and printing test case descriptions have been implemented in Parsifal. Moreover features like document preview and CVS debugging have been added for internal quality assurance. Parsifal's graphical interface consists of three main panes (see Fig. **??**.). Document structure and navigation tree on the top-left. A short description about the meaning of the tree entry is on the lower left, the main pane (content pane) is on the right. The top level tree node is called 'Test Case' and gives the author a detailed summary to which WCAG guideline the test applies. In addition test case meta-data includes information about the WCAG guideline the and success criteria the test covers. This information is relevant for test case authors preparing user tests.

In 'Document Status' field authors can change the document's status (e.g. draft version) and can add a CVS comment to their changes made. Document history and versions are also traceable on this pane. The section labeled 'Files' provides access to relevant test files. Test files are presented in the user panel. On basis of a test file the authors can create test scenarios. Therefore authors can preview test files and read the tests purpose. The test case purpose describes what the test file should test and what the expected test result is. The section 'Required Tests' hosts scenario child nodes describing user tests. Tests can be marked as automatic, end user, experts, and one expert test by the test

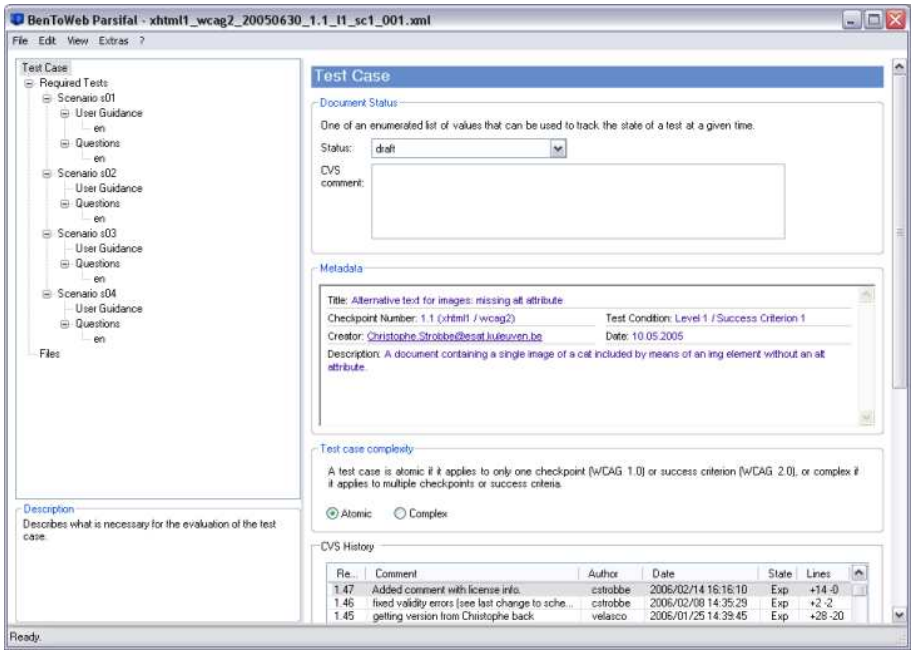

Fig. 1. Status information about a test case description file

case authors. A user test description is only necessary if a test is not marked as automatic test only. Authors can remove and add scenario child nodes as needed. A scenario (see Fig. **??**.) describes user groups to run the test with corresponding questions and user guidance. A user is categorized by his disability/disabilities and his experience level in usage of assistive technologies, user agents and devices. For all three experience levels detailed information about each type (e.g. browser, screen reader), exact product name, product version, and experience level can be specified. Experience levels range from not experienced to very experienced (1-5). One scenario node includes user guidance and questions subsections. User guidance and questions will be presented to users in multiple languages in Amfortas. Therefore localization of all texts is possible in Parsifal. One user guidance entry describes necessary prerequisites a user must undertake to accomplish a test. One sample instruction could ask the user to switch on his screen reader. All user guidance and question texts are described in a subset of XHTML1.0. Therefore a basic HTML editor component eases authoring user guidance and question text entries. Parsifal supports questions of type yes/no, yes/no/open, multiple choice, and likert scale. Each question type supports different settings like likert scale and multiple choice questions need at least labels and values being specified. Open questions need information about layout and spacing to be presented correctly to user panel by Amfortas. As mentioned before all question texts must be translated into different languages,

this happens by adding child nodes to the Questions entry with corresponding language specific labels.

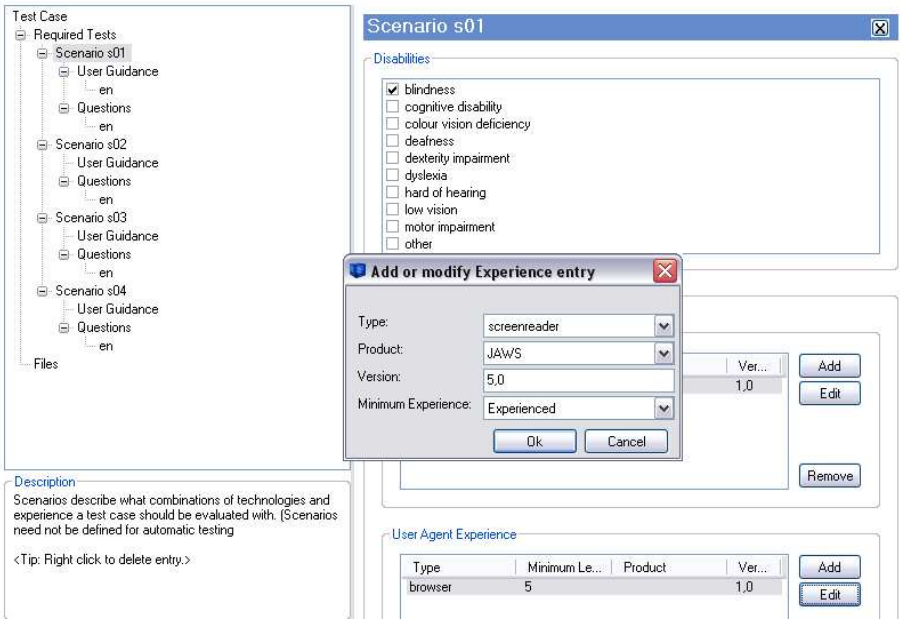

Fig. 2. Scenario pane in Parsifal

On the right hand side question corresponding input can be edited. In this case a yes/no/open question at least needs a question text. A mandatory 'Other Options' entry is used if e.g. yes/no is not an appropriate answer to users. In this example the user was asked to navigate through the page using the tab key and if he got the welcome message. Most of the users will answer with 'yes' or 'no' but those who have e.g. technical difficulties would like to answer more precisely. Therefore 'Other Options' asks for the reason why the user was not able to get the information propagated.

3 Amfortas - Test Case Evaluation Framework

The framework is a Java-based web application for evaluating accessibility test suites. It covers the whole management process from creating and handling user and testing profiles, to storing the test results in a database. Usually user evaluation processes are monitored by an agent, who in a first step presents a task and then gathers information about the process and the result by asking relevant questions. Obviously, this makes testing fairly expensive. The evaluation framework should ease user testing procedures in a way that evaluators can proceed testings by themselves without any human support.

3.1 Process Work-flow

The evaluation task starts with a recruitment procedure. The recruit is guided through a series of questions in order to gather information about his personal constitution (disability, age range, internet experience) and technical equipment (assisitive technology, browser, device used to access web content). The answers of the recruitment procedure determine one testing profile. The evaluator should be able to set-up additional profiles, if for some reason the equipment changes or he uses more than one set of equipment to access web content.

The administrator can view the status and profile of the registered participants. All participants with adequate profile will be granted access to the evaluation framework by activating their accounts. The users then are able to access the log-in area of the web-application, but cant start testing unless the test profile is admitted to a particular test suite.

The testing process starts by activating the corresponding link in the web application. The mapping algorithm first looks up the database, selects profile information for the actual user and tries to match it with the TCDL description files. The matches are stored in a pooling table, in blocks of 20 test cases (called test run). Testing can be repeated as long as there are matching test cases available. The evaluators are expected to have at least moderate English skills. As all test cases are in English, the log-in area of the web-application is also kept in English.

The framework guides the evaluator through all the allocated test cases. A test case is finished when the user answers the question about the test case. The answer is stored in the database.

After evaluating the test suite the data is extracted from the database for later analysis.

3.2 System Architecture

Amfortas is built on top of the XML publishing framework Cocoon[2]. The core of Cocoon's object-oriented architecture is based on the Apache Avalon project. The overall architectural view of Amfortas consists of three components: a Java web server containing the application, a MySQL database and a resource containing the test files and test description files. Usually, the files are provided via a web interface, but any other providing mechanism, e.g. CVS, would also be appropriate.

Due to Cocoon's internal architecture, the evaluation framework is composed out of 3rd party components, own components, Javascript files, XML files and certain additional resources.

Database Layer: Instead of creating a custom persistency layer, we decided to use Hibernate[3]. Hibernate not only provides a powerful and easy-to-use object-

[2] http://cocoon.apache.org/
[3] http://www.hibernate.org

relational bridge for Java applications, but also offers a rich query language to retrieve objects from the database.

The evaluation framework uses 41 tables to store persistent data. Persistent data is data needed to build up the application view, data to accomplish the mapping procedure, data to conduct application management procedures and data which composes the evaluation result. The database model is straightforward: it is actually a normalized view on the users personal condition and technical equipment. Amfortas stores the mapping-related data for assisitive technologies, user agents, devices and disability in different tables, which in the end are consolidated in the table `test_profiles`. One entity set in `test_profile` determines one test profile.

Presentation Layer: The initial version of the user evaluation framework presents a very simple and intuitive user interface, as it is going to be accessed by users with a huge variety of interaction requirements. For later extension of functionality a clear separation of content and presentation is needed.

Amfortas' content has been completely authored in XML reusable entity documents. This process actually involves three sitemap components. If there is no need for aggregation, a Cocoon generator simply loads XML from the file system or web resource and generates SAX events which are handled by consecutive XSLT transformers and finally a serializer (i.e. HTML for Browsers). In most cases, an Cocoon aggregator is required which offers additional functionality by aggregating more than one XML files (i.e. Header, Content and Footer) inside a root element.

Amfortas' public access area is implemented in the languages English and Dutch, as the potential evaluators are recruited in England and Belgium. Cocoon offers the i18n-Transformer component to implement internationalization features. Language-dependent text is stored in an XML file and referenced by the application through a unique key.

CForms - Forms are important for interaction but at the same time raise a lot of accessibility issues. This is mainly due to the need for direct and responsive interaction, which is usually implemented with client-side technologies - in most cases Javascript - which may cause serious accessibility barriers. These problems have been already addressed by W3C, which proposes the next generation of web forms named XForms. XForms seems ideal for Amfortas' forms implementation, but it's not applicable as most user agents have not yet implemented this technology. A good alternative that merits goods from both current and future world are Cocoon forms (CForms). CForms are XML forms that introduce the separation of the model (form model) and instance (form template) of the form that can be implemented separately. The so-called form widgets can be developed and include their own server-side validation. In Amfortas, the form instances are controlled with Cocoon flow. A further advantage of this approach is the ability to move to XForms by simply applying XSL transformation.

Application Logic While 'action components' have been the dominant method to encapsulate application logic in Cocoon, this position has been taken over by the *Control Flow*. A flow script is implemented in Javascript notation. A considerable part of application logic, like the recruitment process, the application administration or the saving routine of the evaluation results is implemented using flow scripts. Application logic which directly influences the view of web application is implemented using the Cocoon JX-Transformer. Higher-level logic, such as the mapping procedure, is implemented in Java classes.

Profile Mapping Only test cases marked with status 'accepted for end user evaluation' are pooled. Each user request for new test cases triggers the mapping algorithm. Mapping involves comparing TCDL disabilities and experience elements (i.e. user agents, assistive technologies and devices) with the user's test profile stored in Amfortas' database. The mapping algorithm first filters out the test cases that are 'done' and those that the user has already evaluated. For a test case to be allocated, the following conditions needs to be satisfied. For disabilities, if in TCDL there is a disability, the test profile needs to have at least one of them. For user agents, assistive technologies and devices, the test profile must have all the types that appear in TCDL file. If the TCDL specifies a product as well, the profiles need to have at least one of the specified products for each type. Further, if minimum level and product version are also specified these need to be equal or less than the profile's one. Finally, for better matching, a complementary grading mechanism is involved that enables a better selection after sorting by grade and getting the required number designated by the test suite configuration.

Test Presentation The test case evaluation is a cyclic process which can be invoked as long as test cases can be allocated for the actual evaluator. The whole test presentation is created within the Cocoon sitemap, by passing a number of parameters from the Cocoon flow, for example, the URI of the actual test case description file, the URI to the test file, and the scenario. The default sitemap generator is used to fetch the test description file from the web resource and page header, footer and navigation from the local file system. The standard cocoon aggregator bundles these XML trees to a single XML tree which is handed over to XSLT transformation and finally serialization. The result is an XHTML page with user guidance information, a link to the test file and a question with corresponding answer type (see Fig. **??**.).

3.3 Application View

Fig. **??**. shows the user interface for the evaluation process. The 'user guidance' section (1) requests the evaluator to adjust special settings or behave in a certain manner in order to complete the test. The question (2) is presented before the link to the test file (3), to give a first idea what to mention when evaluating the test file. Finally the answer section (4) presents one of the answer categories to be replied. On submit, the answer with references to the accomplished test is stored.

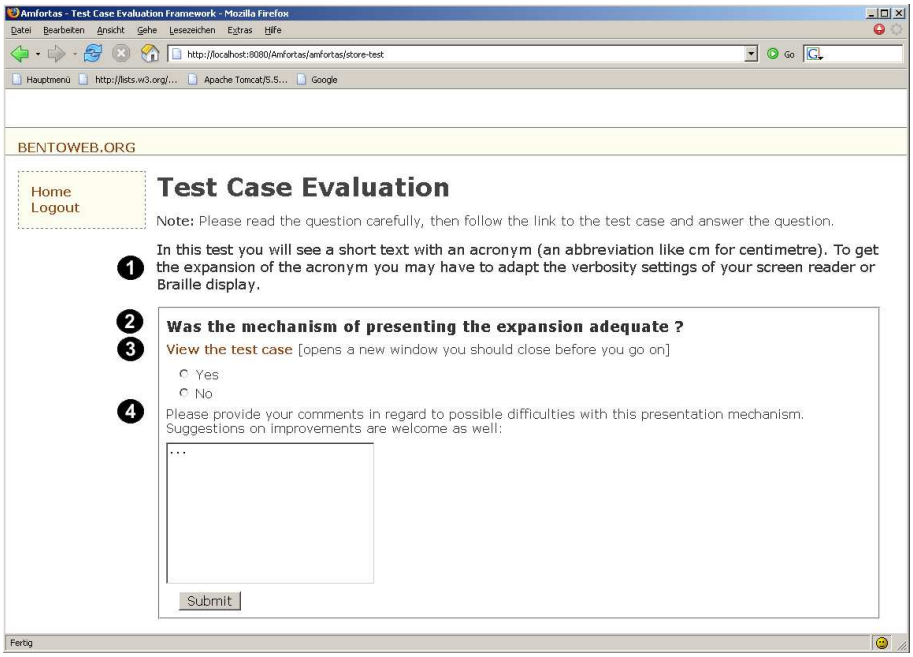

Fig. 3. Amfortas test case evaluation

4 Acknowledgements

This work has been undertaken in the framework of the project BenToWeb IST-2-004275-STP funded by the IST Programme of the European Commission.

References

1. Petri, H., King, N., Gappa, H., Nordbrock, G., Velasco, C.: Large scale evaluation of the accessibility of Web features. Computers Helping People with Special Needs. 10th International Conference, ICCHP 2006, Linz, Austria, 12-14 July 2006, Proceedings.
2. Strobbe, C., Herramhof, S., Vlachogiannis, E., Velasco, .C.: Test Case Description Language (TCDL) - Test Case Metadata for Conformance Evaluation. Computers Helping People with Special Needs. 10th International Conference, ICCHP 2006, Linz, Austria, 12-14 July 2006, Proceedings.
3. Brajnik, G.: Comparing accessibility evaluation tools: A method for tool effectiveness. Universal Access in the Information Society. Springer Verlag (2004), pp. 252-263.