A Proposed Architecture for Large Scale Web Accessibility Assessment

Mikael Holmesland Snaprud[1], Nils Ulltveit-Moe[1], Anand Balachandran Pillai[2], and Morten Goodwin Olsen[1]

[1] Faculty of Engineering, Agder University College, NO-4876 Grimstad, Norway
{Mikael.Snaprud, Nils.Ulltveit-Moe, Morten.G.Olsen}@hia.no
[2] SpikesourceIndia Pvt Ltd, Bangalore - 560034, India
abpillai@gmail.com

Abstract. This paper outlines the architecture of a system designed to demonstrate large scale web accessibility assessment developed in a European research project. The system consists of a set of integrated software components designed to automatically evaluate accessibility metrics for a large number of[1] websites and present results in a common report. The system architecture is designed to be maintainable, scalable, and extensible in order to facilitate further development of the tool. To meet these design criteria within a limited set of resources, an Open Source approach is adopted both for selecting, designing and developing the software.

Keywords: Software architecture, Web accessibility evaluation, free/open source software.

1 Introduction

Access for all to the Information Society is a declared key-goal for the European Union. With the wide-spread adoption of the Internet – in particular for access to government information and services, it is essential to secure access for all citizens. However, there are still digital barriers that make access to information difficult, especially for people with disabilities. Large scale web accessibility benchmarking can help to locate potential barriers and fuel the development towards Internet accessibility for all.

The EIAO (European Internet Accessibility Observatory) [1] project will establish the technical foundations of an Internet observatory for large-scale assessment of the accessibility of European websites.

The general specification of the demonstrator is defined in the Unified Web Evaluation Methodology, UWEM0.5 [2] which is based on WCAG1.0 [3]. To build a demonstrator implementation of UWEM0.5 we need to integrate different web assessment tools and aggregate the results into one coherent report. Experts often use

[1] The project is co-funded by the European Commission DG Information Society and Media, under the contract IST-004526.

K. Miesenberger et al. (Eds.): ICCHP 2006, LNCS 4061, pp. 234 – 241, 2006.

several automatic tools to support their work of evaluating web accessibility. The results are then manually compiled into one report.

In this paper we outline some related work on web assessment components and architecture and elaborate on the chosen software components and interfaces in a proposed architecture. Finally, we discuss potential extended uses of the proposed architecture.

2 Related Work on Tools and Integrating Architecture

Most existing web evaluation tools are built to be self-contained and do not encourage integration with other tools. However, such automated tools often give a better coverage of accessibility aspects when used in combination. In [4] an architecture is proposed for integrating test tools to be used by practitioners who carry out web accessibility testing.

This problem has a range of common issues with the approach pursued in the EIAO project. Those issues include the difficulty to combine the reports from different tools both in terms of format and aggregating results, and the need to run tools individually and sequentially.

The proposed architecture in [4] outlines a common infrastructure that will allow tools to be installed, selected and customized. Those tools would share a common crawler and parser. Furthermore, seven tools are analysed in more depth for WCAG [3] coverage.

To classify the tools a taxonomy of capabilities is proposed. For results, a canonical format is referred to. EARL [5] will be addressed for this purpose as soon as EARL is fully defined by W3C.

In contrast with the architecture in [4], we base the tool interface on EARL and replace the indicated user interface with an import interface into the Observatory for further processing. We also need to make sure the integrated evaluation tools are at least partially conforming to UWEM [2].

Some early versions of our work on an architecture was first presented in [6] and later in more elaborated form dividing into components in [7]. An early presentation of measurements results is available in [8].

3 Outline of the EIAO Software Architecture

The architecture is based on a set of requirements derived from the original project proposal, a user requirement analysis and on the defined project Open Source policy as indicated in the next section.

3.1 EIAO and the Open Source Policy

The project management has decided to use the philosophy of OSS in the project. The policy is inspired by [9], by Bruce Perens. The following are the primary reasons for adopting this open approach:

A. Free access to source code, covering

- Availability: The software should be available free of charge so that others could use it without hindrances for research and improvement.
- Extensibility: The software created by the project should be open so as to allow extensibility by the use of plug-ins, adapters or other extension mechanisms.
- Transparency: The transparency of software used for measurements is essential to allow third party verification of the implementation. Also the software development process should be open and well documented to provide others a chance to verify the development methodology and tools.
- Participation: By defining a policy to use and develop OSS software, the project opens itself to participation from international developers and groups, who develop open/free software. This has helped the project to attract wider developer participation.

B. Free access to assessment results, covering

- Dissemination: Providing free access to the measurement data created by the project will help the project to reach out to a wider audience who can benefit from the results of the research.
- Verifiability: The results should be available to allow third party replication of measurements and verifying the metrics.

The above two aspects are crucial for validity of any measurement program.

C. Licensing. The project aims to provide all software developed by it under the GNU [10] General Public License (GPL). The publications of the project will be provided under the Creative Commons Attribution ShareAlike license. For a plug-in interface part of the software, the LGPL license or GPL with an exception is considered; which will allow the use of plug-ins released under proprietary licenses. The chosen licensing scheme has proved to simplify the collaboration within the project allowing the participants to focus on the research rather than on IPR discussions.

D. Open technologies. In addition to Python [11] for software development and Plone [12] for the project website the project uses a number of other OSS components. Those include PostgreSQL, Subversion, Apache, Trac, 3store, HarvestMan, RELAX-NG/ Schematron/EARL and Openoffice.org.

EIAO reuses OSS components to reduce development lead time. The OSS development style allows the changes made to the components to be contributed back to the original projects.

The architecture is based on this policy and designed to meet the functional requirements.

3.2 Outline of the Architecture

The architecture is designed to facilitate the integration of existing and new accessibility evaluation tools into the observatory framework. The main components

are shown below in Figure 1. A cron job will start the controller at regular intervals (monthly). The controller will then load the plugin configuration from the RDF repository, and start binding local and remote WAM components to the Scheduler/Dispatcher, based on the evaluation to be performed on the defined list of URLs in the URL directory.

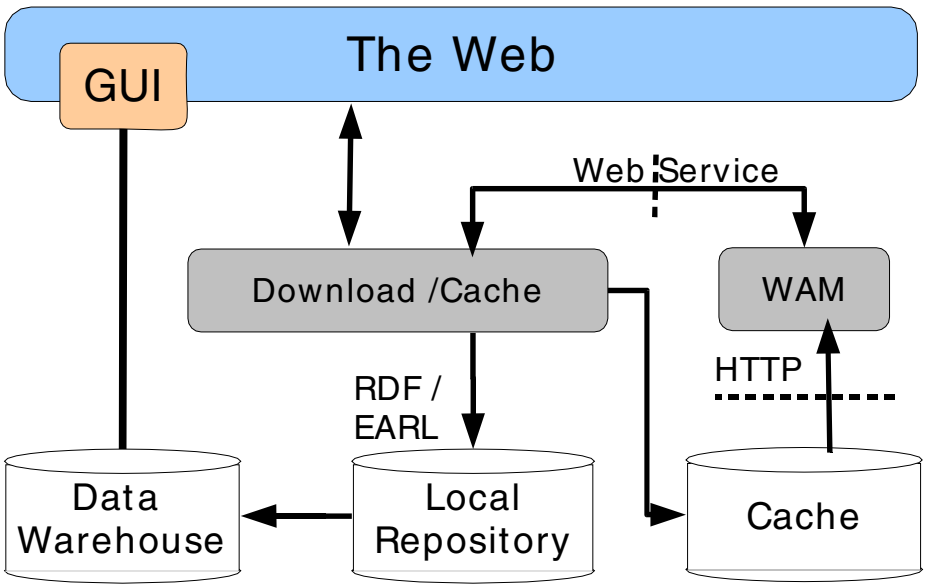

Fig. 1. The architecture of the Observatory planned for the first release

For the implementation of the architecture we will use the HarvestMan crawler [13], Relaxed [14] and Imergo [15] as basis for WAM plugins. In addition, we use 3Store on MySQL [16] as RDF repository, and PostgreSQL [17] as the basis for the Data Warehouse.

Information flow in the observatory. The information flow is structured as follows:

- The crawler downloads web page samples ta the http-cache repository.
- The downloaded web pages get their accessibility measured by internal/external WAMs.
- Subsequently, the sampling algorithm will choose the next sample to download. In release 1, this will be a uniform scheduling algorithm, but the scheduler will be candidate for a more advanced resource allocation algorithm in further releases.
- The results from the WAMs will be processed in a parallel queue-thread and later stored in the local RDF repository. At the same time the crawler will start downloading a new page / sample site.

Writing to the RDF repository and downloading from the Web are potential bottlenecks in the system. An advantage of having the information flow of the

Observatory as described above, is that the two bottlenecks of the system can run in parallel to improve the overall performance of the Observatory.

3.3 The Main Software Components Used in the First Implementation

Crawler. The key component in this architecture is the crawler, which is based on the Open Source web crawler HarvestMan [13]. HarvestMan was chosen because it is a good and mature web crawler that was easy to adapt to the system. Moreover, the HarvestMan developers were interested in actively supporting the EIAO project. HarvestMan has already been adapted to crawl according to EIAOs URL directory by the developer. HarvestMan has been further modified and integrated into the system to perform accessibility assessments on each downloaded web site and to store key use scenarios [2]. These scenarios are estimated by using the near uniform random sampling algorithm [18]. The web content is stored in a 3Store for further processing by web accessibility test modules.

Datawarehouse. A data warehouse is used in the architecture to support storage of large amounts of data and support flexible analysis from an on-line user interface. We also expect the hierarchical design of the databased schema in a data warehouse to facilitate analysis of the data.

PostgreSQL was chosen since it is the only Open Source DBMS with extensibility, table partitioning and bit-mapped indexes. Materialized view support is being built by the bizgres project. PostgreSQL performs very well for complex queries on large databases. The only other Open Source alternative is MySQL, which does not have these features.

Local repository. This component is responsible for storing the downloaded web pages, the URLs to crawl and also the results of the evaluation. The RDF repository will use the 3store RDF triplestore, and we will develop an EIAO RDF schema that can be used as an overall container for timestamped test runs containing EARL metadata, the URL directory, HTTP header information and other metadata extracted from the documents. Also data that does not fit nicely into EARL, like unique descriptions of the HTML deviations from Validate will be stored in the EIAO RDF schema. A formal representation of the EIAO RDF result repository also elaborated. The observatory in version 1 uses only one RDF repository, in order to make extraction of data as simple as possible for the data warehouse ETL tool.

The choice of a triplestore for the RDF repository was influenced by the co-operation with W3C on developing the EARL report language, since a large-scale EARL/RDF repository, able to collect data from different vendors, will be the first large-scale EARL consumer application. This may also be a driving force for vendors to use EARL.

Web Accessibility Metrics (WAMs). A Web Accessibility Metric (WAM) is a formal rule specifying how to make a statement about accessibility barriers of a given Web resource.

The description of a WAM consists of one or more procedures for how to measure characteristics of a Web resource and one or more procedures using the results of the

measured characteristics to produce probability values for the violation of accessibility requirements derived from WCAG 1.0 / UWEM0.5.

In addition, the WAM will have one aggregation scheme (according to UWEM) that defines how to aggregate the probability values to produce a statement about an aspect of accessibility barriers of the Web resource. The current implementation gives two aggregated scores indicating the barrier probability and the change in barrier probability since the most recent measurement.

A set of Web Accessibility Metrics (WAMs) has been specified based on the tests described in UWEM0.5, checking for deviations from web standards (WCAG guidelines [3]) based on WCAG 1.0 checkpoints, with a view to conversion to WCAG 2.0 if possible. The first set of WAMs will be based on Relaxed/Schematron [14]. The first WAMs will address simpler forms of rules that can be expressed as Schematron rules. More complex rules may be implemented in some other language in a later release.

Cache. The Observatory must implement a web page repository to store versions of documents downloaded during the test runs and associated versioned tests and other metadata in raw form. It is important to store data unmodified to verify correct operation of the Observatory and as evidence for the data being tested.

Another important feature that should be considered is, facilities to avoid storing duplicates of the mirrored web documents. If the individual documents can be verified, by using digital signatures or hashing, we could save disk space by storing only a single version of each document downloaded. This is an important consideration, since storing 12 separate copies of test runs with all the (X)HTML data will exceed the current storage capacity of the Observatory.

To facilitate the checking of documents via WAM components or external tools, it is important that the web cache can be accessed via a transparent proxy, since most existing accessibility assessment tools will load missing web resources autonomously using the HTTP protocol. This will also apply to some of our own design bases, like the Relaxed validator.

4 Future Development of the Architecture

We have proposed a design suitable for integrating already existing software components to reduce the development and maintenance effort. We expect the architectures to evolve as the project matures, which should help to improve the performance. Interesting still unresolved issues include how to deal with the integration of potentially conflicting results from different tools or how to strike a balance between accuracy storage demands and sampling.

To make sure that the measurements are actually measuring barriers in a meaningful way, we have planned an improvement cycle involving systematic user testing. The test will be carried out with different disability groups on a set of web sites also assessed automatically. We plan to experiment with pattern recognition techniques and learning automata to identify the relations among user experience and automatically assessed web content. We also plan to improve the WAMs by extensive

user feedback, to keep them updated with evolving Web technology. A more elaborate description of this approach is given in [19].

5 Conclusion

The proposed architecture builds on the results from a number of Open Source projects to establish a demonstrator for an Internet Accessibility Observatory.

Much effort was initially spent on defining the requirements and on code analysis to select the most suitable code bases. For the first implementation of the architecture we use the HarvestMan crawler [13], Relaxed [14] and Imergo [15] as basis for WAM plugins. In addition, we use 3Store on MySQL [16] as RDF repository, and PostgreSQL [17] as the basis for the Data Warehouse.

The results from the prototype implementation indicate that adapting and integrating existing tools rather than building a system from scratch has been very suitable for the task. This choice has also, implemented by the highly talented development team, resulted in an extensible architecture for large scale web accessibility evaluation.

We think that a similar architectural approach can be useful also in related research projects and plan to explore this potential further in future research.

References

1. European Internet Accessibility Observatory - http://www.eiao.net
2. Unified Web Evaluation Methodology version 0.5 - http://www.wabcluster.org/uwem05/
3. Web Content Accessibility Guidelines version 1.0 - http://www.w3.org/TR/WAI-WEBCONTENT/
4. Englefield, P., Paddison, C., Tibbits, M., Damani, I.: A Proposed Architecture for Integrating Accessibility Test Tools, IBM Systems Journal, Special Issue on accessibility, Vol 44, No. 3, (2005) 537-555, - http://www.research.ibm.com/journal/sj/443/englefield.html
5. Evaluation and Report Language - http://www.w3.org/TR/EARL10-Schema/
6. Snaprud, M.H., Jensen, C.S., Ulltveit-Moe, N., Nytun, J.P., Rafoshei-Klev M. E., European Internet Accessibility Observatory (EIAO), eGEP - Getting full Impact of eGovernment - launching a European Measurement Framework-, Vienna, Austria, (2006)
7. Ulltveit-Moe, N., Gjøsæter, T., Snaprud M. H.: An Open Architecture for Large Scale Monitoring of Web Accessibility - Workshop on Web Accessibility and Metamodelling, Grimstad Norway, (2005)
8. Snaprud, M.H., Ulltveit-Moe, N., Granmo, O.C., Menk, M.E., Wiklund, A., Sawicka, A.: Quantitative Assessment of Public Web Sites Accessibility - http://www.etsi.org/cce/proceedings/7_3.htm
9. Perens. B.: Global Technology Policy Institute - http://techp.org/
10. Stallman, R.: GNU General Public License (GPL) - http://www.gnu.org/copyleft/gpl.html
11. van Rossum, G.: Python - http://python.org/

12. Plone: A user-friendly and powerful open source Content Management System - http://plone.org/
13. Pillai, A.B.: HarvestMan (2005) - http://harvestman.freezope.org/
14. Nalevka, P.: Relaxed HTML Validator2005 - http://badame.vse.cz/validator/
15. Velasco, C.: Imergo 2005 - http://www.fit.fraunhofer.de/projekte/imergo/index_en.xml
16. Riddoch, A., et al: 3store2005 - http://sourceforge.net/projects/threestore/
17. PostgreSQL Global Development GroupPostgreSQL2005 - http://www.postgresql.org
18. Henzinger, M.R., Heydon, A., Mitzenmacher, M., Najork, M.: On Near-Uniform URL Sampling (2005), http://research.compaq.com/SRC/mercator/papers/www9/
19. Craven, J., Snaprud, M.H.: Involving Users in the Development of a Web Accessibility Tool, http://www.ariadne.ac.uk/issue44/craven/intro.html