Architecture for large-scale automatic web accessibility evaluation based on the UWEM methodology.

Nils Ulltveit-Moe[1], Morten Goodwin Olsen[2], Anand B. Pillai[1], Christian Thomsen[3], Terje Gjøsæter[1], Mikael Snaprud[1]

[1]Faculty of Engineering and Science,University of Agder, 4876 Grimstad, Norway, email: Nils.Ulltveit-Moe@uia.no, abpillai@gmail.com, terje.gjosater@uia.no, mikael.snaprud@uia.no

[2] Tingtun AS, Kirkekleiva 1, 4790 Lillesand, Norway, email: morten.g.olsen@tingtun.no

[3] Dept. Of Computer Science, Aalborg University, 9000 Aalborg, Denmark, email: chr@cs.aau.dk

Abstract

The European Internet Accessibility Observatory (EIAO) project has developed an observatory for performing large scale automatic web accessibility evaluations of public sector web sites in Europe. The architecture includes a distributed web crawler that crawls web sites for links until either a given budget of web pages have been identified or the web site has been crawled exhaustively. Subsequently, a uniform random subset of the crawled web pages is sampled and sent for accessibility evaluation. The evaluation results are stored in a Resource Description Format (RDF) database that later is loaded into the EIAO data warehouse using an Extract-Transform-Load (ETL) tool. The aggregated indicator results in the data warehouse are finally presented in a Plone based online reporting tool. This paper describes the final version of the EIAO architecture and outlines some of the technical and architectural challenges that the project faced, and the solutions developed towards building a system capable of regular large-scale accessibility evaluations with sufficient capacity and stability. It also outlines some possible future architectural improvements.

Keywords: large scale, web accessibility evaluation, web crawling.

Introduction

The European Internet Accessibility Observatory (EIAO) project [1] is an IST/STREP project[1], that has developed an online web accessibility observatory as a tool for large scale automatic web accessibility evaluations of public sector web sites in Europe[2].

The EIAO Observatory has a flexible architecture that has the opportunity to play a significant role for large scale automatic benchmarking comparing some elements of web accessibility across countries in Europe. This can give the stakeholders an indication of the current status and trends between individual countries or other geographical locations, which means that the evaluation results from the EIAO Observatory may be used as part of a framework to get feedback on how well the anti-discriminatory laws [2] [3] and proposed procurement accessibility requirements [4] within EU/EFTA work.

The web sites evaluated have been selected by Capgemini in the 20 Services Web Sites Evaluation [5], and are representative of each evaluated country. The URLs include public web sites for job search, building permission, police, public libraries, education and public procurement.

[1] The EIAO project is an Information Society Technology (IST) Specific Targeted Research Project (STREP) co-funded by the European Commission DG Information Society and Media under the contract IST-004526.

[2] The EIAO Observatory can be used to perform automatic web evaluations for any web site, however the EIAO project has focused on a set of web sites covering common public services used throughout Europe.

This paper was presented at the NIK 2008 conference. For more information see http://www.nik.no/

The architecture has been designed to meet a set of requirements concerning both performance and statistical properties of the evaluation results. The requirements with largest impact on the software architecture are:

- Evaluate at least 2500 sites monthly.
- Sample uniformly from a set of URLs from each web site.
- Handle non-validating HTML.
- Quick response time for queries to the data warehouse.

This paper describes the final version of the EIAO architecture and outlines some of the technical and architectural challenges that the project has dealt with. It also describes the chosen solutions towards building a system architecture capable of regular automatic large-scale web accessibility evaluations meeting the statistical requirements and the requirements for scalability, capacity, and stability.

Related work

Several commercial web accessibility evaluation toolkits that provide automatic web crawling functionality exist, e.g. HiSofts AccVerify [6], UsableNet LIFT [7] and Imergo [8]. Most of these tools are built to support web quality management in the enterprise, and therefore focus primarily on providing detailed information about accessibility deviations that can be used for improving web accessibility. These tools can also be used for more thorough web accessibility assessments for individual web sites including both automatic and manual testing by experts.

EIAO has a slightly different objective, since it focuses on performing large-scale automatic web crawling over geographic areas, to calculate an accessibility indicator that makes it possible to compare accessibility results among web sites, countries or regions and also present accessibility indicator trends over time.

Most existing accessibility surveys have been carried out on a national level - for example the annual quality survey of public web sites in Norway done by norge.no [9], while a few surveys exist on European level [5]. Existing surveys are to a large extent based on manual evaluation of web sites, which is labour intensive. Also the methodologies for evaluating public web sites are not harmonised, which means that it is difficult to compare accessibility surveys from different European countries [10]. EIAO aims at filling this gap by providing frequent and relatively inexpensive automatic web accessibility measurements based on the Unified Web Evaluation Methodology (UWEM 1.2) [11]. Even though the amount of accessibility problems that can be tested automatically is limited[3], it is believed that frequent measurements showing aggregated results and trends for an accessibility barrier indicator in a region is useful for the stakeholders.

Outline of the large scale accessibility evaluation architecture

In this section, we briefly describe the Observatory's components. The complete description can be found in the EIAO source code documentation [12]. The pipeline of operations when evaluating web pages are indicated by the arrows in the Observatory architecture shown in Figure 1.

To start an evaluation, the set of sites to crawl is loaded into the Site URL server. A set of web crawlers, monitored by a crawler server, fetches URLs for web sites to crawl and then crawls the web site in a breadth-first manner[4] until 6000 URLs have been

[3] UWEM 1.2 has for instance only declared less than 20% of the total number of tests as automatable.

[4] A breadth-first search is a graph search that begins at the seed URL and explores all the neighbouring nodes, before it then explores the next level of the graph. This in contrast to a depth-first search that traverses the web graph recursively.

identified or the entire set of visible web pages has been scanned, whichever comes first.

When a sufficiently large set of URLs have been identified and stored in the URL repository, the crawler tells the sampling server that it can start sampling the web site. The sampling servers starts the sampler process, which extracts a uniform random sample of 600 pages (selection without replacement) from the set of web pages in the URL repository. Note that the architecture allows running multiple samplers in parallel. This sample is then sent to the Web Accessibility Metrics (WAM) server for accessibility evaluation, and the resulting Evaluation And Reporting Language (EARL) [13] report is stored in an RDF [14] database. When the sampling is finished, the RDF database is sent to the Extract-Transform-Load (ETL) server, which starts a process that loads the evaluation results into the data warehouse that is optimised for fast read access.

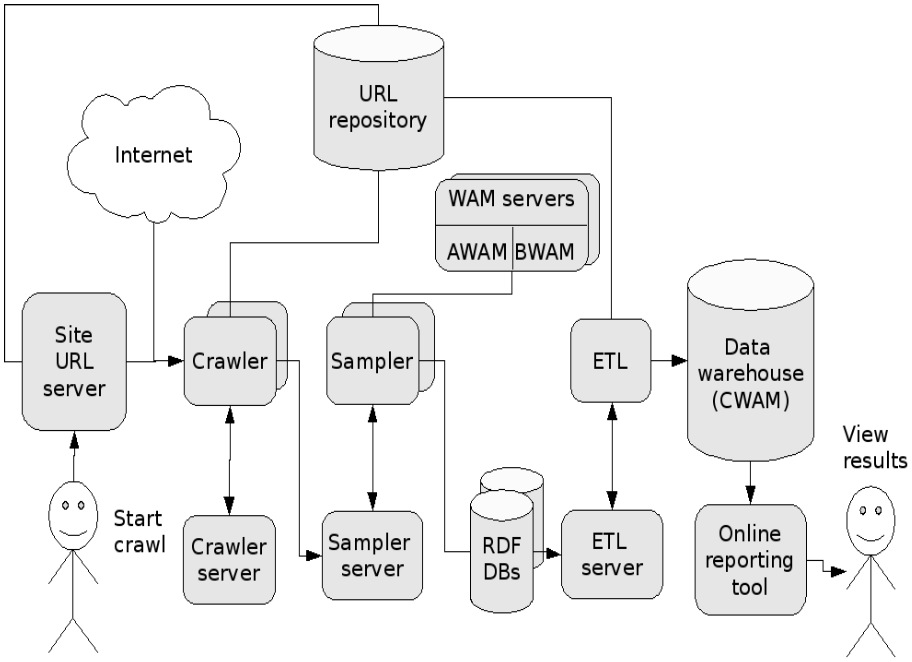


Figure 1: EIAO version 2.2 architecture.

After all web sites have been crawled, sampled and loaded into the data warehouse, final tasks like building indexes and filling materialised views in the data warehouse are done. The data warehouse is now ready to be accessed by the online reporting tool.

Discussion of current architecture

The EIAO architecture has improved in several iterations over the last three years. The Observatory is now mature, stable and can run large crawls repeatedly with little maintenance. The current architecture for large-scale evaluation of web accessibility aims primarily at providing good quality statistics about the evaluated web sites. Good quality statistics ideally require that the web sites are scanned exhaustively to identify all web pages and that a sufficiently large uniform random sample from all web pages is being evaluated. For practical reasons, which will be discussed in subsequent sections, we have had to loosen the requirement of scanning entire web sites for URLs by introducing a stop criterion.

Distributed web crawler

The EIAO crawler is based on the open source web crawler HarvestMan [15], which is a Python based multi-threaded web crawler. The EIAO project has written a crawler server that maintains a set of HarvestMan crawler processes, where each process samples one web site. Each crawler process runs with 10 parallel threads fetching data from the web site being crawled and 10 parallel threads crawling the fetched web pages using a breadth-first strategy. The web crawler uses memcached [16] to store information about the web pages that have been visited within a web site. In addition, internal crawler queues are stored on disk when they exceed a given size to crawl large web sites without using too much memory. The complete EIAO machinery is able to load on average 0.7 pages per second, using 2 crawler machines and 6 WAM servers for accessibility evaluation[5]. This means the EIAO has got capacity to crawl and evaluate 100 web sites daily using the current crawling strategy on the current hardware.

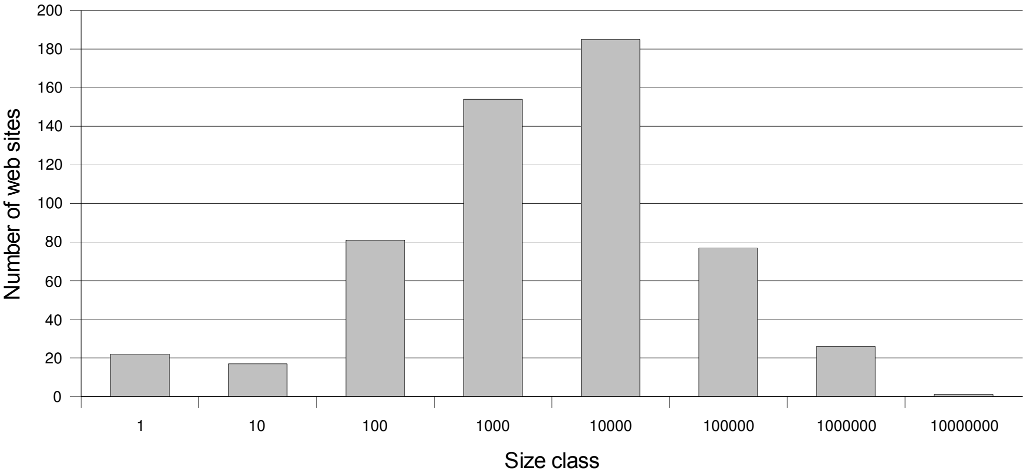


Figure 2: Distribution of web site size of selected public web sites.

To evaluate whether an exhaustive scan based sampling would be viable or not, we carried out an experiment to determine the distribution of web sites sizes in the set of public URLs we were testing. This estimate was based on a uniform random sample of 609 sites from the set of URLs used [5]. We first determined their size via a set of Google queries for a specific site or domain[6]. We assumed that Google would be able to give a reasonable estimate of the visible part of these web sites. The visible parts of the web are the web pages that are not password protected or otherwise hidden behind forms or other protection mechanisms. 563 of the sampled web sites were indexed by Google, and are part of the calculations shown in Figure 2 and Table 1.

Google is currently able to index most common file formats [17], and it is also able to crawl Flash based web sites [18]. It is however not yet able to fully crawl JavaScript and other technologies apart from Flash and (X)HTML. The latest crawl of 568 sites from Germany, Norway and EU-level shows that JavaScript is used extensively, with on average 9.8 occurrences per HTML page [19]. This means that the visible part of the web can be expected to be somewhat larger than the Google based estimate indicates. During the test phase of the EIAO observatory, we found that 3.6% (114 out of 3208) of the sites from the Capgemini 20 services set of URLs were impossible to crawl by the EIAO Observatory due to extensive use of unsupported technologies on the home page.

[5] The hardware used for the large crawls was mostly 4 CPU Intel Xeon based servers with 2.3 and 3 GHz CPUs. One of the WAM servers and one of the crawlers was only a 2 CPU 3 Ghz machine. All servers were running Fedora Core 5. The data warehouse server was a 4 CPU Intel Xeon server with 3GHz CPU and two fast SCSI disks with in total 3Tb disk space in RAID5 configuration. Each disk had 160 Mb/s transfer speed.

[6] Google supports querying for a specific site by using the site: selector. E.g: site:.eiao.net The search results will then show the number of URLs that matches this site.

The Google estimate should therefore give a useful indication of the distribution of web site sizes the EIAO Observatory can expect to find.

Figure 2 shows the distribution of web sites across size classes. The number of sites in each size class is shown on the y-axis. The size classes are denoted by their upper limit in a logarithmic scale. For example, the size class 1000 contains all sites having between 101 and 1000 pages. The distribution seems to be log-normal with the largest size class for web sites between 1001 and 10000 pages. The median, as shown in table 1 is 1160 pages, which also is within the largest size class.

Largest number of pages indexed by Google	4260000
Average number of pages indexed by Google	22857
Median number of pages indexed by Google	1160
Number of sites with less than 6000 pages	423 (75%)

Table 1: Results from experiment to analyse web site size.

Our test report [20] shows that one crawler machine has a speed of 2.75 pages/second on our hardware. Note that this is significantly more than the evaluation speed of the entire EIAO Observatory[7]. This section describes the crawling which is only part of the pipeline of operations in the Observatory.

Running with two crawler machines gives us a performance of downloading 5.5 pages per second[8]. This means that a dedicated crawler server would use 18 days to crawl the largest web site in the sample with 4.26 million pages. This is still within the one month window that the EIAO has to perform a large scale evaluation. However, the relatively few large web sites will take too much of the processing capacity to complete all web sites in time. Assume that we need to evaluate 3000 web sites with an average number of 22857 pages with two crawlers managing 2.75 pages/second. In this case, it would take 144 days to complete the crawl, which clearly is outside the one month limit as set by project requirements. Additionally, the pages need to be evaluated for accessibility barriers, loaded into the data warehouse etc. The conclusion is that the exhaustive scan approach is not viable, so we need some kind of stop criterion.

Since it is not viable with our resources to crawl the web sites exhaustively, the crawling strategy (for instance, breadth-first, depth-first, or random walk) will also have an impact on the end result, since only part of the web site will be crawled. Najork and Wiener [21] have shown that a breadth-first search is a good crawling strategy, since the most important web pages, measured as the web pages with highest page rank, are detected early in the crawl. In practice, this means that web pages with many in-links[9] are detected early in the crawl. We therefore chose a deterministic breadth-first strategy to ensure repeatability. This strategy will also limit the influence from some very deep web structures, like for example unbounded web calendars that otherwise would cause a bias in the end result. The selection of algorithms and indicators are further explained in the UWEM Indicator Refinement document [22].

[7] The Observatory throughput is currently limited by the accessibility evaluation capacity, which is the reason why most of the machines are dedicated to accessibility evaluation.

[8] It is technically possible to run with more crawlers, however the crawl was limited to two crawlers to give more machinery for performing sampling and accessibility evaluations in order to ensure that the entire pipeline of operations was optimised for providing as good performance as possible with the available hardware within the required one month time interval. It was in fact the accessibility evaluations and not the crawlers that were the limiting factor for overall throughput.

[9] The number of in-links to a web page is the number of URLs pointing towards that web page.

We chose to limit the scan to stop when 6000 URLs have been identified, This means that 75% of all web sites will be crawled exhaustively. The sample size of 600 web pages was chosen to achieve a precision of 0.02 at 95% confidence level, based on an estimated standard deviation of 0.25 and on the measured capacity of the WAM servers.

Decoupling the sampling and crawling completely and using a relatively large, uniform random sample of 600 samples is a statistically sound method that also was easy to implement and understand. The downside of this technique is that to achieve a sufficiently good crawler performance, a quite aggressive crawling strategy was implemented, using 10 threads in parallel towards the web site being crawled. One way to mitigate this problem, is to perform a broader crawl, using more web crawlers, with each web crawler crawling the web site less aggressively. This does not reduce the total amount of web pages that need to be downloaded, but it reduces the instantaneous load of the web server being crawled. Another possible strategy would be to perform the scan for URLs more seldom and re-sample using URLs from previous scans. The problem with this approach, is that it is oblivious to changes in the structure of the web site, like new web pages or web pages with changed URLs. However updated web pages where the URLs have not changed will still be part of the sample. Cho et al [23] have shown that on average 20% of the available web pages change on a monthly basis. It is therefore not obvious that re-sampling using the existing set of URLs can be performed, so further experiments would have to be done to verify if this is viable.

Another possible solution would be to do experiments to verify if a near-uniform random walk solution gives sufficiently good results to be usable in practice. The random walk based algorithms should perform better if they are being run using the results from a large URL scan as seeds. However a random walk based algorithm would not reflect the freshness of the web site well, as discussed in [22]. Another possibility might be to do a more targeted search, by for example doing a breadth-first search of the 600 first URLs and then evaluate the web pages these URLs referenced. This would clearly be deterministic and repeatable. The problem is that such a sample only would give a statement about the accessibility of the part of the web that had been scanned and selected, which means that only a small part of the web site could claim to be covered by the accessibility indicator. The current strategy will on the other hand provide a strong claim about the accessibility indicator for the entire part of the web site scanned, which as before mentioned is the entire web site in 75% of the cases and a significant part of most remaining web sites.

The easiest optimisation is therefore to start with a broader crawl on a larger set of web sites, for instance by assigning one web site to each crawler thread instead of running with 10 threads towards one web site. The other alternatives can be verified experimentally later, if needed.

Sampling and evaluation

The sampling server is separated from the crawler server as a stand-alone service, so that sampling and crawling are completely decoupled. The sampling server is multi-threaded, and supports sampling of different web sites in parallel[10]. After the crawler has identified 6000 URLs or the complete web site, the web site is passed on to the sampling server responsible for managing individual sampler processes. The sampler process extracts a uniform random sample of 600 pages[11] from the set of pages

[10] In order to utilize the CPU of the WAMs, five sampling server threads are run in parallel on one the sampling server machine.

[11] The sampling server will sample 600 samples without replacement. If there are less than 600 pages in the web site, then the entire web site will be sampled. Further note that a page may consist of several URLs when a web site is created with frames.

identified and stored in the URL repository. Each of the selected URLs will then be downloaded and sent to the Web Accessibility Metrics server (WAM) for evaluation.

The WAM server is a multi-threaded server that receives SOAP requests [24] to evaluate a web page, and returns an EARL report [13] with the evaluation result. (X)HTML processing is based on the Relaxed HTML validator [25], which uses the Schematron/RNG [26] rule processing engine. The main components of the WAM server, are a set of Analytical WAM modules (AWAMs) that parse the HTML and extract parameters that are used in the accessibility evaluation and a set of Barrier computing WAMs (BWAMs) that use the AWAM parameters to produce statements about accessibility barriers.

The AWAM that handles CSS stylesheets is implemented in a similar way as a web browser, i.e. by parsing the (X)HTML document for references to CSS and performing CSS cascading calculations for each HTML element, to decide which CSS rules that apply to the HTML element. Only applicable CSS rules are checked by the AWAM module, presuming a default media type of screen. For example, only blinking text that would be visible on a web browser should count as a barrier indicator. CSS validation is done at the same time as the CSS rules are parsed by the Batik CSS SAC parser [27].

The results from the Schematron evaluation, validation and CSS evaluation are sent to the BWAMs for UWEM barrier indicator computations and also to extract content statistics from the web page (e.g. content type). The BWAM results are currently either 0, indicating no barrier or 1 indicating barrier. All BWAM results are then converted to an Evaluation And Reporting Language (EARL) report, which is returned to the Sampler, which then calculates the accessibility score of the site *f(s)* based on the evaluation results. The UWEM accessibility indicator score for a web site *s* is defined as the fraction of all failed tests to the total number of applied tests:

$$f(s)=\frac{\sum_{p\in S} B_p}{\sum_{p\in S} N_p}$$

Where *f(s)* is the UWEM score for site *s,* B_p is the number of failed tests for web page *p,* N_p is the number of applicable tests for web page *p* and *S* is the set of sampled web pages. For example, assume a web site S with one page *p* that has $B_p=3$ failed UWEM tests and N_p=9 applicable tests, then the UWEM score value will be *f(s) = 3/9 = 1/3,* which means that 1 out of 3 automatable UWEM tests indicated a barrier.

When the complete sample has been evaluated, the RDF database is sent to the ETL server for loading into the data warehouse. Further aggregation over several web sites, is done in the data warehouse by calculating the average score value over the web sites:

$$f(G)=\frac{1}{N}\sum_{s\in G} f(s)$$

Where *f(G)* is the UWEM indicator for a set of web sites *G,* and *N* is the total number of web sites in *G*.

The sampling server will then continue with the next web site to sample. Currently, the WAM accessibility evaluation is the limiting part of the processing pipeline, even though only around of 10% of the identified URLs are evaluated. The most time consuming operations in the WAM server are currently CSS and Schematron processing. The Schematron processing has been optimised by caching the parsed XSL Transformation (XSLT) [28] stylesheets, so that only the first web page evaluated requires 3 pass XSLT processing to parse the Schematron and only one XSLT transformation is required on the second and subsequent runs. A CSS cache implemented as a patch to the Batik parser also caches the parsed stylesheets to improve the speed of CSS handling. The evaluation speed has been compensated by running several WAM servers in parallel. However, this increases the hardware costs to run the EIAO software.

The main problem with the current sampling approach, is that the large fixed sample size requires quite high computational load to perform the accessibility evaluations using the set of WAM servers.

One possible approach to reduce the number of samples, would be to use adaptive web sampling suggested by Thompson [29]. The main problem with this approach, is that aggregating over several web sites is problematic from a statistical point of view, since the amount of samples varies significantly between the different web sites. This optimisation is therefore not viable, as discussed in [22].

Another possible WAM optimisation is to change the scheduling, so that the sampler selects one WAM to do all evaluations for a given web site, instead of using the current strategy of passing the requests to the WAMs in a round-robin fashion. This could improve the performance of the CSS handling, since the probability of a CSS cache hit would increase.

Data warehouse and Extract-Transform-Load (ETL)

To perform analysis of the collected accessibility data in an easy, reliable and fast manner, the data warehouse EIAO DW has been built. In the EIAO DW, data at a very low granularity is stored. For example, it is represented that a specific subject (e.g., an element of an HTML page) at a specific time on a specific date was tested with a specific WAM implementation which gave a specific result. But also "contextual information" such as the URL of the (X)HTML document, its size, language, etc. is stored. Although the EIAO DW holds data at a very low granularity, end-users will only use reports that aggregate the data at higher levels and, for example, show accessibility results for entire web sites or countries.

In the EIAO DW, a *star schema* [30] is used. A simple star schema is, however, not enough and the schema has thus been extended with so-called *outriggers* and *bridge tables* (see [30]) to be able to represent the complex web data. With the generic schema used in EIAO DW, it is possible to add new WAMs by inserting a row and without changing the database schema. For details about the EIAO DW schema, see [31].

Like the rest of the Observatory, the EIAO DW has been implemented only using open source software. PostgreSQL version 8.x [32] has been chosen as the database management system (DBMS) due to its maturity, extensibility and features.

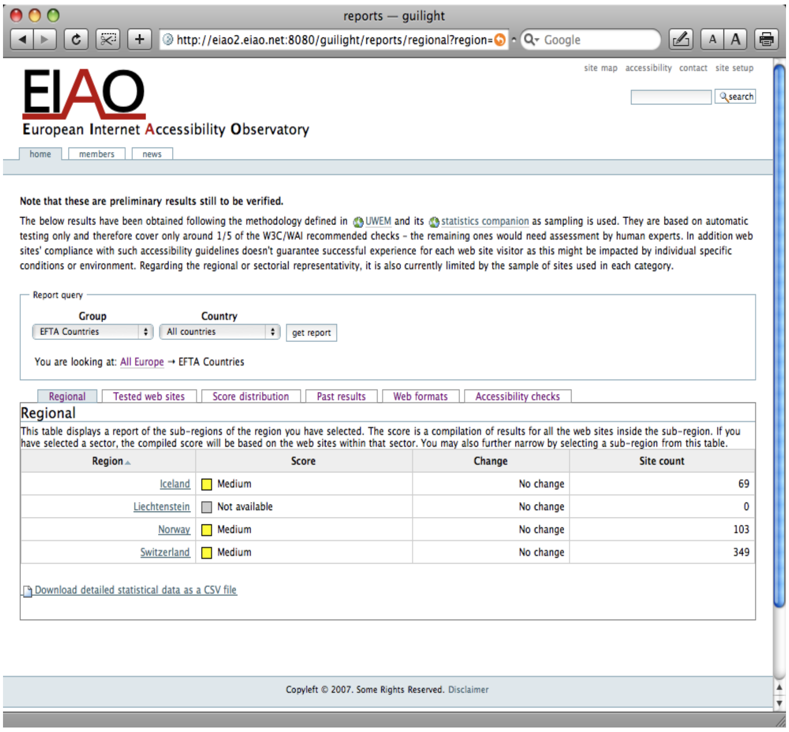
When the EIAO DW is loaded with data, a hand-coded Extract-Transform-Load (ETL) application is used. No existing open source ETL application was found suitable for this purpose. The Sampler and WAMs generate RDF and EARL data which conceptually takes the form of a graph. The ETL tool then has to traverse this graph and load the data into the data warehouse. For large RDF structures, using a traditional RDF triplestore was time consuming. To speed up the ETL process, we designed a domain specific RDF database that uses traditional regular expressions to parse the EARL. The ETL load process then runs in a pipeline where one thread reads the entire RDF database into memory and the other thread uses an already read, in-memory database to transform the EARL data and load it into the data warehouse.

In addition, the ETL tool uses bulk loading of all data that is not queried during the ETL process. This means that instead of inserting the resulting rows one-by-one, the ETL can write the data to a file which is then loaded (in a fast manner) at the end of the process. Further, the ETL to a large degree uses main memory for caching values to avoid look-ups in the database. For efficient memory usage, memcached [33] is used for the larger RDF structures.

The on-line reporting tool described in the following section presents (aggregated) data from the EIAO DW. When the reporting tool queries the EIAO DW, stored procedures in the database are used. This gives us a large degree of flexibility: It is possible to do updates (both to the code and the database schema) without the reporting tool being affected as long as the interface to the stored procedures remains unchanged. This choice has proven itself useful for the project several times. For example, it was previously a problem that some of the reporting tool's reports took too long time to compute. To solve this problem, some *summary tables* were created and filled with pre-aggregated data. The relevant stored procedures could then be updated to use these summary tables and the performance improved significantly. In the current 2.2 release of EIAO DW, 10 such summary tables exist to boost the performance. The summary tables are filled as the last step during the load of the EIAO DW. Due to the large amounts of data, it has been important to carefully fine-tune the SQL code that fills these summary tables such that the job is done reasonably fast.

For the current data volumes, the EIAO DW and ETL tool scale well enough. If the data volumes later get so big that the used solution does not scale well enough anymore, it could be considered to distribute the DW over several servers. If, for example, it becomes a problem to load the DW within a reasonable time-frame, it could be considered to have a separate DW for different groups of countries, e.g. Nordic, Western European, Eastern European, and Southern European countries. These four servers could then be loaded in parallel. However, some changes would then be needed for the reporting parts of the Observatory since results from several DWs would have to be combined to form the final result. Other possibilities include using some of the existing replication mechanisms for PostgreSQL. If, for example, the Observatory later in time gets so many reporting requests that they cannot be answered quickly enough, a setup with replication and read-distribution can be introduced.

On-line reporting tool


reports — guilight
http://eiao2.eiao.net:8080/guilight/reports/regional?region=
Google
EIAO
European Internet Accessibility Observatory
site map
accessibility
contact
site setup
search
home
members
news

Note that these are preliminary results still to be verified.

The below results have been obtained following the methodology defined in UWEM and its statistics companion as sampling is used. They are based on automatic testing only and therefore cover only around 1/5 of the W3C/WAI recommended checks - the remaining ones would need assessment by human experts. In addition web sites' compliance with such accessibility guidelines doesn't guarantee successful experience for each web site visitor as this might be impacted by individual specific conditions or environment. Regarding the regional or sectorial representativity, it is also currently limited by the sample of sites used in each category.

Report query

Group: EFTA Countries
Country: All countries
get report

You are looking at: All Europe → EFTA Countries

Regional

This table displays a report of the sub-regions of the region you have selected. The score is a compilation of results for all the web sites inside the sub-region. If you have selected a sector, the compiled score will be based on the web sites within that sector. You may also further narrow by selecting a sub-region from this table.

Region ▲	Score	Change	Site count
Iceland	🟨 Medium	No change	69
Liechtenstein	⬜ Not available	No change	0
Norway	🟨 Medium	No change	103
Switzerland	🟨 Medium	No change	349

Download detailed statistical data as a CSV file

Copyleft © 2007. Some Rights Reserved. Disclaimer

Figure 3: The EIAO Online Reporting Tool.

The EIAO Online Reporting Tool (Figure 3) is based on Zope and Plone, and is implemented as a Plone product. It can display different reports on the data in the data warehouse:

- *Regional report* (default) displays a list of sub-regions within the selected region, with Region name, Score, Change since previous result, and Number of sites in the geographical group. The Region name is clickable, taking you to results for that sub-region. The Score column displays simplified scorecards indicating the result with colour and text.
- *Tested web sites report* shows a list of the tested web sites within the selected region, with scorecards and change information as for the Regional report, as well as number of sampled pages.
- *Score distribution report* shows a score distribution bar chart for the selected region. There is one bar for each of the possible scores, and the height of each bar represents the percentage of sites in the chosen region that achieved that score. Below the chart is also a list of the different scores with the percentages of each.
- *Past results report* shows a list of past results for the selected region. It shows the evaluation date, score and change from previous test runs, and also number of sites in the selection.
- *Web formats report* show statistics about different types of web formats of sites within the selected region.
- *Accessibility checks report* shows detailed UWEM test results for the selected region. We show the UWEM test name, grouped by WCAG checkpoint, and the percentage of tests that failed for this UWEM test. The percentage of failed tests are displayed both numerically and as a filled bar.

The Online Reporting Tool has its own internal cache for values read from the data warehouse. This increases the performance of read access to the Online Reporting Tool even further.

Conclusion

The Observatory is now mature, stable, produces statistically sound results and works well for accessibility evaluations of up to 2500 web sites monthly using the available hardware. There are however two main problems with the current approach. Firstly, the crawlers currently use a crawling strategy that is too focused on too few servers by running 10 crawler threads in parallel towards each web site. This puts a significant strain on the the measured web servers, because the measurement is not evenly spread out in time. Secondly, the current approach requires a large portion of a web site to be downloaded, but only 10% of that information is used in actual accessibility evaluations. The current approach is therefore not very bandwidth efficient in terms of accessibility information extracted per megabyte of data.

To mitigate the first problem, the processing load on the web servers being evaluated can be lowered much by ensuring that the crawling of each web site is more evenly spread out in time. The easiest way to achieve this, is to modify the crawler so that each crawler thread operates on a different web site. The lowest possible instantaneous load could be achieved if each web site got one thread dedicated that spread its fixed number of web pages to download evenly over the one month interval of a crawl by inserting appropriate time delays. The lowest possible load would however not be attainable, because the sampling of the web sites and loading of web sites into the data warehouse would be poorly distributed in time. However it should still be possible to lower the load significantly from what we experience today by performing a broader crawl than we currently do.

To mitigate the second problem, improving the bandwidth efficiency of the EIAO, can only be done by doing methodological changes. The current sampling methodology can be expected to be close to the goal of extracting a uniform random sample from a given web site. It can therefore be used as a “gold standard” against which other sampling methodologies can be tested. We could therefore run a set of tests where different strategies were tested out like: reduced frequency of URL scanning and re-sampling using the existing set of URLs scanned, direct sampling using random walk or using a deterministic breadth-first search of only the first 600 web pages. If empirical evidence shows that any of these alternative approaches only deviate insignificantly from the “gold standard”, then this alternative, more bandwidth-efficient approach should be chosen in favour of the current approach.

The WAM evaluation process can also be enhanced by improving the scheduling of the requests sent to the WAMs so that we can get better performance from the CSS cache. This can be done to force each Sampler to only use one WAM server and further run as many samplers in parallel as there are WAM servers available.

Bibliography

1: Mikael Snaprud, European Internet Accessibility Observatory, accessed 2008, http://www.eiao.net

2: The council of the European Union, Establishing a general framework for equal treatment in employment and occupation, 2000, http://europa.eu/scadplus/leg/en/cha/c10823.htm

3: Mark Bell, Extending EU Anti-Discrimination Law: Report of an ENAR Ad-Hoc Expert Group on Anti-Dicrimination Law, 2008, http://ec.europa.eu/social/main.jsp?langId=en&catId=89&newsId=373&furtherNews=yes

4: European Telecommunications Standards Institute, Human Factors (HF);European accessibility requirements for public procurement of products and services in the ICT domain, 2008, http://portal.etsi.org/stfs/STF_HomePages/STF333/ETSI%20DTR%20102%20612%20v50%20Sep24.doc

5: Patrick Wauters, Graham Colclough, Online Availability of Public Services: How Is Europe Progressing?, 2006, http://ec.europa.eu/information_society/eeurope/i2010/docs/benchmarking/online_availability_2006.pdf

6: HiSoftware inc., AccMonitor, accessed 2008, http://www.hisoftware.com/access/newmonitor.html

7: Usablenet inc., Lift machine, accessed 2008, http://www.usablenet.com/usablenet_liftmachine.html

8: Fraunhofer Gesellschaft, Imergo Online [BETA], accessed 2008, http://imergo.com/home

9: Norge.no, Kvalitetsvurdering av offentlige nettsteder, Accessed 2008, http://www.norway.no/kvalitet/

10: Snaprud, M., A. Sawicka, Large scale web accessibility evaluation - a European perspective, in Universal Access in Human-Computer Interaction. Applications and Services, 4556, Springer Berlin/Heidelberg, 2007, pp. 150-159.

11: Eric Velleman, Colin Meerveld, Christophe Strobbe, Johannes Koch, Carlos A. Velasco, Mikael Snaprud, Annika Nietzio, D-WAM4 Unified Web Evaluation Methodology (UWEM 1.2 Core), 2008, http://www.wabcluster.org/uwem1_2/

12: Nils Ulltveit-Moe, Morten Goodwin Olsen, Anand B. Pillai, Terje Gjøsæter, EIAO documented source code, 2008, http://www.eiao.net/publications/EIAO_D5.2.2.1-2.pdf
13: Daniel Dardailler, Sean B. Palmer, Evaluation and Report Language (EARL) 1.0 Schema, 2007, http://www.w3.org/TR/EARL10-Schema/
14: Ivan Herman, Ralph Swick, Dan Brickley, Resource Description Framework (RDF), 2004, http://www.w3.org/RDF/
15: Anand B. Pillai, HarvestMan - The HarvestMan Web Crawler, accessed 2008, http://www.harvestmanontheweb.com/
16: Brad Fitzpatrick, memcached, accessed 2008, http://www.danga.com/memcached/
17: Google, Frequently Asked Questions - File Types, 2008, http://www.google.com/help/faq_filetypes.html
18: Google blog, Google learns to crawl Flash, 2008, http://googleblog.blogspot.com/2008/06/google-learns-to-crawl-flash.html
19: EIAO Observatory, EIAO Web Formats, Accessed 2008, http://eiao2.eiao.net:8080/guilight/reports/contentstatistics?
20: Morten Goodwin Olsen, Nils Ulltveit-Moe, Annika Nietzio, Mikael Snaprud, Anand B. Pillai, Test report for EIAO version 2.2 of the Observatory software, 2008, http://www.eiao.net
21: Marc Najork and Janet L. Wiener, Breadth First Crawling Yields High Quality Pages, 2001, http://www10.org/cdrom/papers/pdf/p208.pdf
22: Annika Nietzio, Nils Ulltveit-Moe, Terje Gjøsæter, Morten Goodwin Olsen, UWEM Indicator Refinement, 2008, http://www.eiao.net/publications/Indicator_refinement_FINAL_v1.31.pdf
23: Junghoo Cho, Hector Garcia-Molina, The Evolution of the Web and Implications for an Incremental Crawler, 1999, http://citeseer.ist.psu.edu/419993.html
24: W3C, SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), 2007, http://www.w3.org/TR/soap12-part1/
25: Petr Nalevka, Relaxed, the HTML validator, accessed 2008, http://relaxed.vse.cz
26: ISO, Information technology - Document Schema Definition Languages (DSDL) Part 3: Rule based validation - Schematron, 2006, http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip
27: Apache Software Foundation, Batik SVG Toolkit, 2008, http://xmlgraphics.apache.org/batik/
28: James Clark, XSL Transformations (XSLT)Version 1.0, 1999, http://www.w3.org/TR/xslt
29: Steven K. Thompson, Adaptive Web Sampling, 2006, http://www.stat.sfu.ca/~thompson/research/aws-finaldraft.pdf
30: R. Kimball, M.Ross, The Data Warehouse Toolkit 2nd edition, 2000
31: Torben Bach Pedersen, Christian Thomsen, Functional specification and architecture of EIAO DW, 2008, http://www.eiao.net/publications/EIAO_D6.1.1.1-2.pdf
32: PostgreSQL Global Development Group, PostgreSQL, Accessed 2008, http://www.postgresql.org/
33: Brad Fitzpatrick, memcache, accessed 2008, http://www.danga.com/memcached/