Tools for Anomaly and Failure Detection in Web Applications

Filippo Ricca and Paolo Tonella
ITC-irst,
Centro per la Ricerca Scientifica e Tecnologica.
38050 Povo (Trento), Italy.
{ricca, tonella}@itc.it

Web applications have become crucial components of our life. However, research studies claim that the experienced quality and reliability of Web applications is often poor or not satisfactory. Evaluation tools are expected to give an important contribution to the improvement of Web applications. Unfortunately, the functionalities offered by available (commercial and research) tools are often limited.
Aim of our previous work was to define and implement two tools (ReWeb and TestWeb) able to provide some novel, interesting analyses and testing techniques.
In this paper, ReWeb and TestWeb are presented and used to find some anomalies and failures in four case studies. At the end of the paper a list of analysis and testing tools for Web applications is considered and confronted with ReWeb and TestWeb.

Keywords: Web application quality, evaluation tools, anomaly and failure detection, multilingual Web site consistency, keyword based clustering, structural Web testing.

In the last few years, Web applications have become of crucial interest for companies working in the commercial and information fields, as well as for a host of social and educational institutions. The importance and economic relevance of Web applications is before everybody's eyes. For this reason, they are subjected to a high pressure to deliver and to a very short time to market. To keep these complex systems working decently, Web programmers are forced to move quickly to the implementation phase and to painfully patch the signalled defects. Despite the promises of new formalisms and tools for Web development, the quality of existing Web applications is in general poor. Authors refer to this situation as the "Web crisis" [1].
To improve the quality and reliability of Web applications, programmers should be supported by automated **evaluation tools** – tools that detect anomalies/failures and return a report or a rating – to be used *before* delivering the initial version of the Web application, but also during its *evolution* over time.
The functionalities offered by available tools range from syntax checking to link validation and accessibility verification. Frameworks and libraries are also available for the definition of spiders (to download all the pages of a Web site and possibly represent them graphically) and for the specification of test cases that can be executed automatically. Only a few research prototypes have been designed to support structural

testing of Web applications or to reverse engineer a high level model of their organization.
We have implemented the two tools **ReWeb** and **TestWeb** to support some new analysis techniques and structural testing of Web applications. We focused our research on a subset of all possible anomalies and failures that can be found in Web applications. The anomalies considered in our work are associated with the structure of the Web applications (e.g. navigation problems), with the structure of the Web pages (e.g. frame errors and clone analysis) and with multilingual Web applications (e.g. missing translations and hyperlinks inconsistency). Our testing approach finds failures, i.e., deviations from the expected behavior.
Using our tools, we have observed a proliferation of Web-based systems developed "naïvely" and containing a lot of anomalies and failures. A critical review of our empirical work [5], started in 2000, shows that just 40% of the Web applications randomly selected for our study are free of anomalies and failures.
Moreover, there are big opportunities for tools to discover such defects and anomalies automatically or semi-automatically, guiding the user towards the verification steps. Thus, evaluation tools are expected to give an important contribution to the improvement of the quality of Web applications.

ANALYSIS AND TESTING TOOLS: *ReWeb* and *TestWeb*

Analysis can be exploited to check whether a Web application satisfies suitable constraints and to detect possible anomalies, while *testing* can be used to expose possible failures, i.e., deviations of the application from the intended behaviour.
ReWeb implements various specific Web application analyses while **TestWeb** supports the Web programmer/tester in the testing phase. The roles of our tools are schematized in Figure 1. Diamonds represent user inputs, circles outputs and ellipsis manual activities. Persistent data are graphically depicted with the usual icon.
The role of the user in the process of anomaly and failure detection in Web applications is very important. The user has to:

- insert some inputs to drive **ReWeb** and **TestWeb** in the phases of downloading and testing;
- execute manual interventions to add information to the model;
- interpret results of analysis reports to determine the anomalies;
- inspect TestWeb's output pages to assess whether the test cases are passed or not.

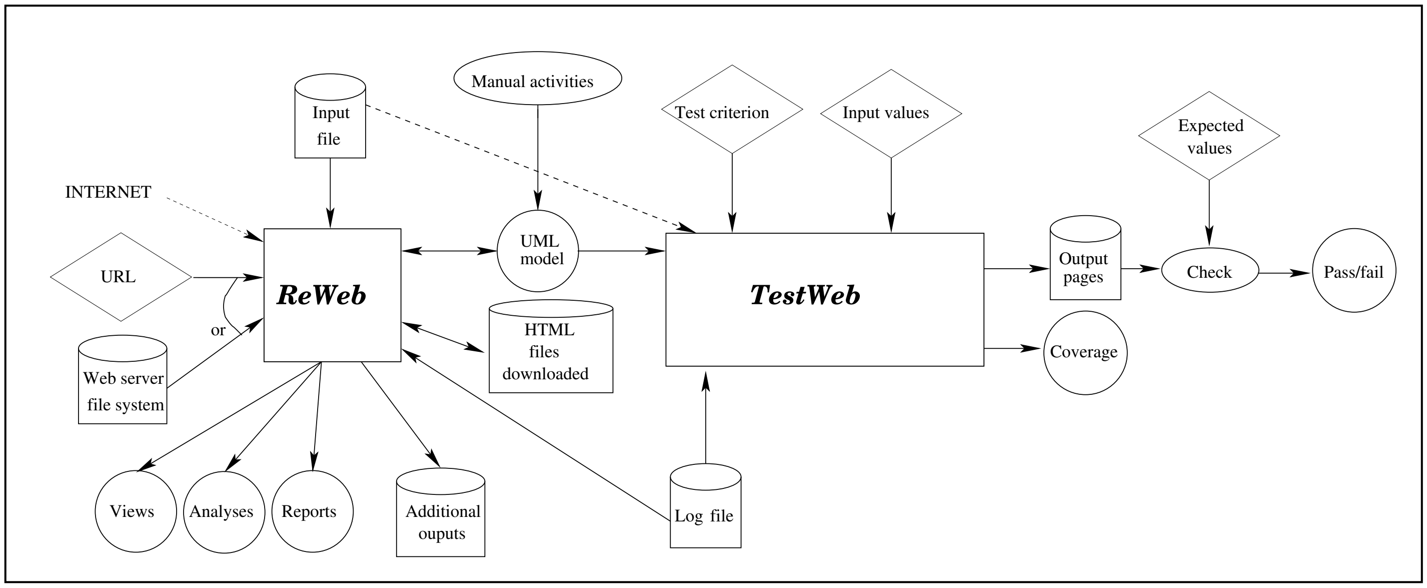


Figure 1: *ReWeb and TestWeb.*

ReWeb

ReWeb [2] contains a spider that downloads the pages of the target Web application with the purpose of building a model and producing views and analyses.

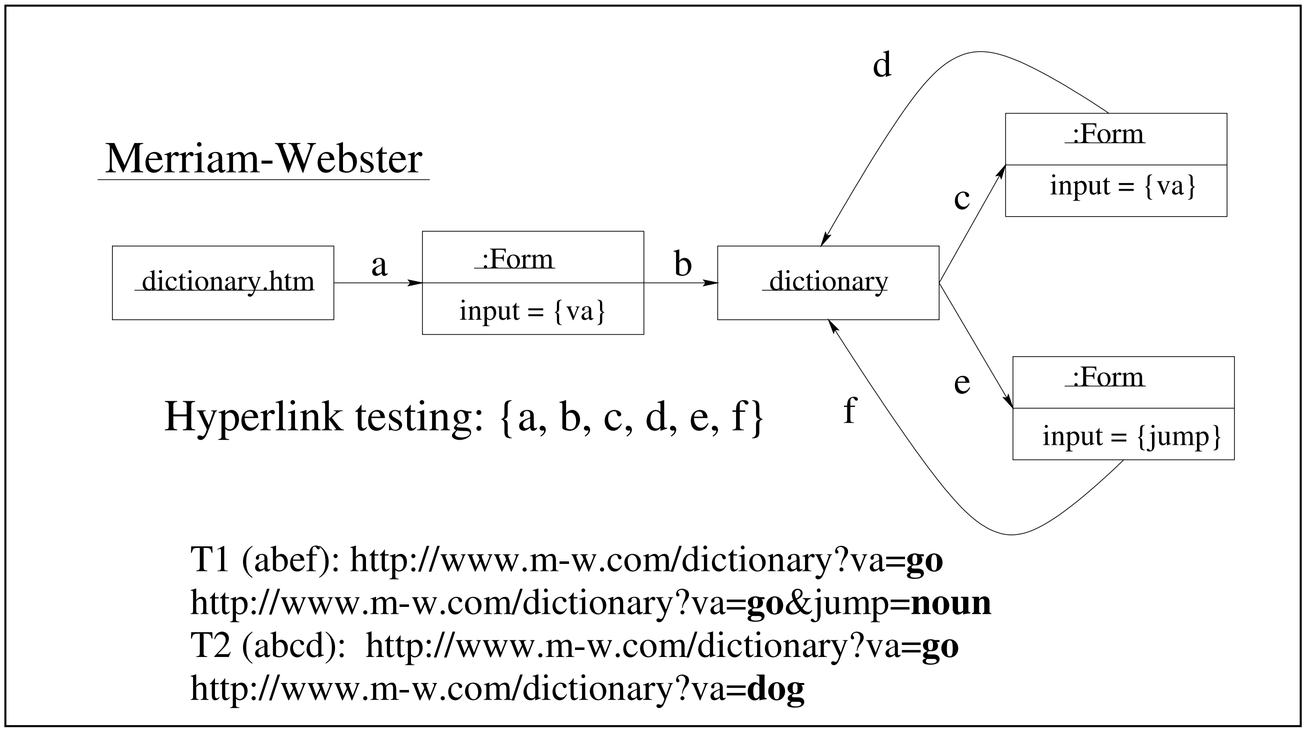
ReWeb implements two main views: the **structural view** and the **system view**. In the structural view, nodes correspond to Web pages, while edges represent hyperlinks between them. The system view is more abstract than the structural view. In fact, nodes represent directories and edges exist only if there is a page in a directory connected to a page in another directory.

Our tool implements also a large set of analyses. Some are simple, as for example the **unreachable pages** (pages available on the server but not reachable from the initial page) and the **ghost pages** (pages associated with "pending links"), others are more complex. One of them is the analysis of the **reaching frames** that aims at determining the set of frames in which a page may appear. The result of the reaching frames analysis is useful to understand the assignment of pages to frames. The presence of *frame errors* is made clear by this analysis. Examples are the possibility to load a page at the top level, while it was designed to always be loaded inside a given frame, or the possibility to load a page inside a frame where it should not be. Another typical situation is the presence of a recursively infinite subdivision of a page into frames. This occurs when a page split into frames can appear inside one of its frames if a proper navigation path is followed. Since it divides itself into the same frames present at the top level, the same subdivision is recursively replicated inside the nested frames.

Some analyses, such as the **multilingual Web site consistency** and the **multilingual hyperlink analysis,** apply only to multilingual Web applications, i.e., applications where the information is supplied in more than one language (English, Italian, French, etc.). This kind of Web applications is more complex than traditional monolingual applications and presents further quality challenges. In particular, the presented

information as well as the site structure should be "consistent" across the different languages. Not only the available information should be the same and should be displayed with the same format, but hyperlinks should comply with the overall site organization, leading to the requested information in the right language.
Another analysis provided by ReWeb is aimed at identifying the static Web pages that are clones of each other (**clone analysis**). The practice of replicating the HTML structure of the pages may lead to problems similar to those known in software maintenance in the presence of code clones [4]. The typical case occurs when "a change" in an HTML page has to be propagated to all its clones in the given Web site.
Keyword based clustering is used in ReWeb to group together pages with similar content. The presence of common keywords is exploited to decide when it is appropriate to group pages together. This analysis can be useful to reveal a reorganization of the Web application pages into more meaningful groups.

TestWeb
The basic idea behind structural testing is that one cannot trust a given piece of software if it contains parts that were never executed during testing. In structural testing the internal structure of a Web application is accessed to measure the coverage that a given *test suite* (collection of test cases) reaches, with respect to a given *test criterion* (e.g., page and hyperlink coverage). A *test case* for a Web application is a sequence of pages to be visited plus the input values to be provided to the pages containing forms. Therefore, it can be represented as a sequence of URLs specifying the pages to request and, if needed, the values to assign to the input variables (see Figure 2, bottom).
TestWeb produces a set of paths [3] from the model produced by **ReWeb** according to the user defined testing criterion. Once paths from the model are generated, the user has to insert input values for the variables collected through forms.
Figure 2 shows the model of the Merriam-Webster Web application as produced by **ReWeb**. In the case of the hyperlink testing criterion, **TestWeb** generates the paths T1 and T2 (bold values excluded), which together ensure link coverage. Input values such as "go", "noun" and "dog" are inserted manually by the user.



d
Merriam-Webster
:Form
input = {va}
c
dictionary.htm
a
:Form
input = {va}
b
dictionary
e
:Form
input = {jump}
f
Hyperlink testing: {a, b, c, d, e, f}
T1 (abef): http://www.m-w.com/dictionary?va=go
http://www.m-w.com/dictionary?va=go&jump=noun
T2 (abcd): http://www.m-w.com/dictionary?va=go
http://www.m-w.com/dictionary?va=dog

Figure 2: *Hyperlink testing for the Merriam-Webster Web application.*

Then, **TestWeb** automates the execution of the test cases. Once **TestWeb** has downloaded the pages associated with each execution, the Web tester can inspect them to assess whether the test cases are passed or not.

ANOMALIES

The quality of the Web applications is a complex, multidimensional attribute that involves several aspects. Among them, the most important ones are the correctness, reliability, usability, accessibility, security, performance and conformance to standards [7]. We focused our research on a subset of all possible anomalies that can be found in Web applications. The anomalies/failures considered in our work are showed in the Table 1.

Anomalies	**Description**
Defect in high level design	When a portion of a WA is unreachable or the directories subdivision is not meaningful
Navigation problem	When a page is not reachable or a pending hyperlink exists
Frame error	When a page fails to divide itself into frames or an inconsistent page loading is present
Missing translation	(only in Multilingual WAs) When the translation of a page is missing
Unintended language switch	(only in Multilingual WAs) When a link causes an unintended switch between languages
Hyperlink inconsistency	(only in Multilingual WAs) When the hyperlink structure in different language sub-sites is not the same
Deviation from expected behaviour	When the expected page is different from the obtained page
Cloned pages	When the HTML structure of a page is replicated

Table 1: *Types of anomalies considered.*

Below, we summarize 4 case studies (others are presented in [5]) that we have conducted in the last 5 years during our empirical assessment of the quality of existing Web applications.

UBICUM

Figure 3 shows the system view of the Web site UBICUM (2001 version) that hosts an Internet provider. This simple static site (86 HTML pages) is decomposed into eight virtual sites, physically associated with different directories, serving different clients, and a directory, *bin*, containing a program to compute the Web server statistics.

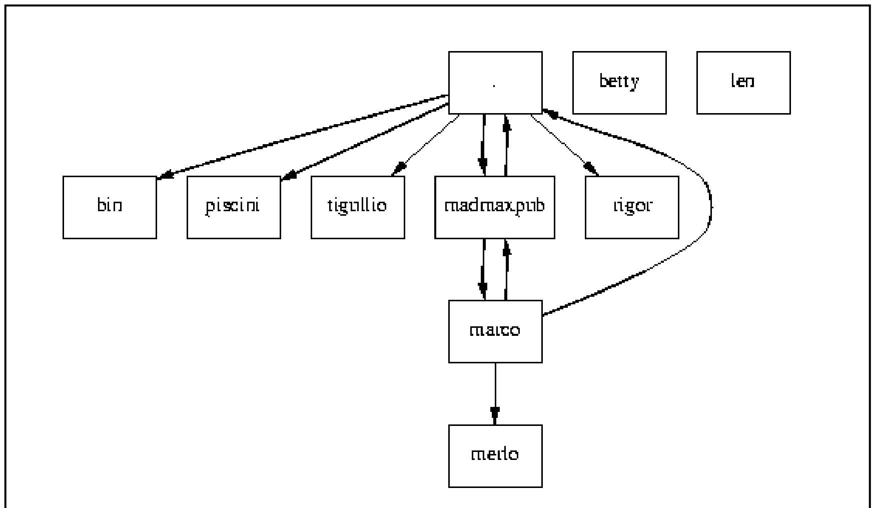
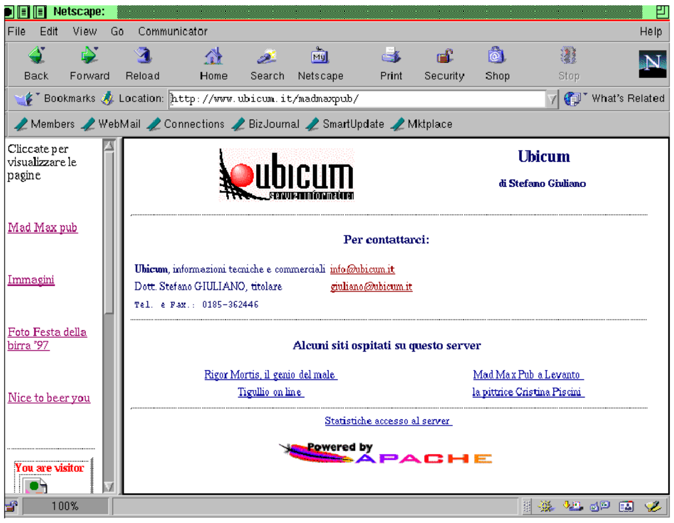


Figure 3: *System view of UBICUM*

This Web site presents three kinds of anomalies: *defect in high level design*, *navigation problem* and *frame error*. UBICUM has a *defect in high level design,* because the virtual sites *betty* and *len* are not reachable from the root of the site (represented as a dot in the System view). UBICUM has also navigation problems. These problems are highlighted in the report with the "pending links" and can be revealed by inspecting the structural view (connectivity among pages is very high). Some hyperlinks are ghosts and the pages of *Rigor* contain two similar menus, with more or less the same links (this can be confirmed by looking at the structure of the Web pages). One is implemented in pure HTML, the other one in Javascript. A reasonable conjecture is that the Javascript menu was added after a restructuring intervention, thus doubling, in some cases, the links between pages. The reaching frames analysis applied to UBICUM showed a *frame error* anomaly. This anomaly is apparent in Figure 4, where the information displayed on the right is not indexed by the menu on the left (Frames are not aligned). The effect of this anomaly could be the creation of a potentially dangerous recursive loop.


Netscape:
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://www.ubicum.it/madmaxpub/ What's Related
Members WebMail Connections BizJournal SmartUpdate Mktplace
100%

Cliccate per visualizzare le pagine

Mad Max pub

Immagini

Foto Festa della birra '97

Nice to beer you

You are visitor

Ubicum

di Stefano Giuliano

Per contattarci:

Ubicum, informazioni tecniche e commerciali info@ubicum.it
Dott. Stefano GIULIANO, titolare giuliano@ubicum.it
Tel. e Fax.: 0185-362446

Alcuni siti ospitati su questo server

Rigor Mortis, il genio del male
Tigullio on line
Mad Max Pub a Levanto
la pittrice Cristina Piscini

Statistiche accesso al server

Powered by APACHE

Figure 4: *Inconsistent page loading: the madmaxpub menu and the main page are displayed concurrently*

IEN

Multilingual Web site consistency and *multilingual hyperlink analysis* have been applied to the official Web site of the National Electro-technical Institute (IEN) ``Galileo Ferraris" (2002 version). This Institute performs important duties in the metrological field regarding time and frequency, electromagnetic, photometric and acoustic quantities. IEN is a medium bilingual site (857 HTML pages for a total of about 165 kLOC), providing the same information in Italian and English. The initial page contains two links to the Italian and English version respectively, and (almost) the same page organization is replicated within the two sub-sites.

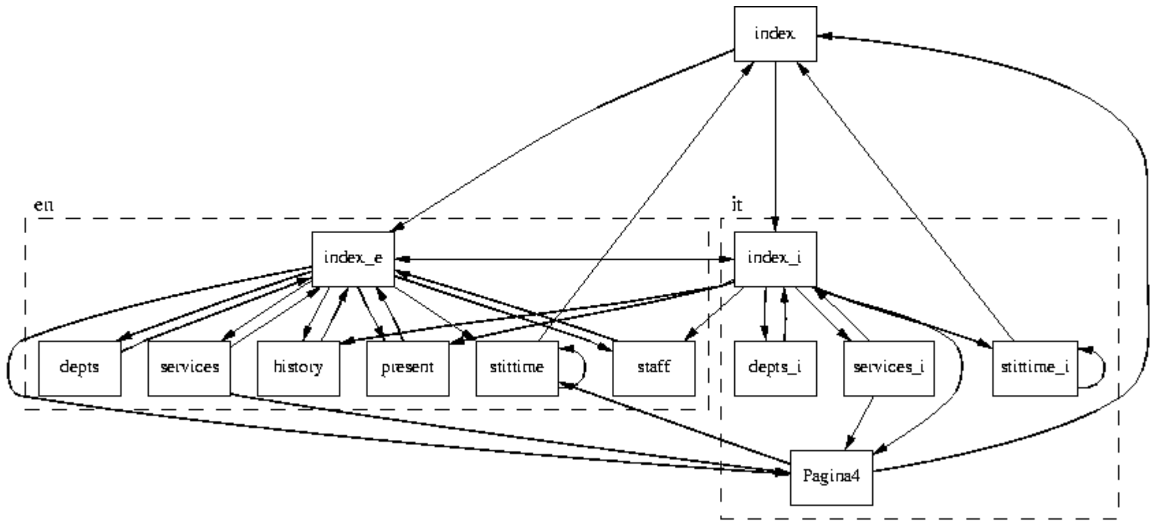


Figure 5: *Structural view of a portion of the bilingual site IEN.*

A visual inspection of the portion of IEN reported in Figure 5, confirmed by our multilingual analyses (Figure 6), immediately reveals the following inconsistencies: *missing translation* and *unintended language switch*.

Page matching:
index_e <–––> index_i
depts <–––> depts_i
services <–––> services_i
stittime <–––> stittime_i

Missing translations:
{index, history, present, staff, Pagina4}

(possible) Unintended switch:
Pagina4 ––> stittime

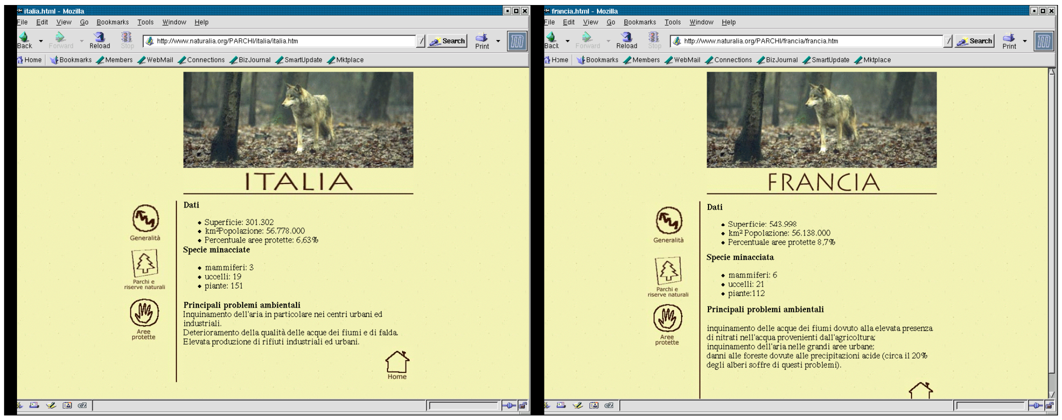
Figure 6: *Multilingual Web site consistency report.*

Page *Pagina4.html* is available only in Italian, but it is referenced also by English pages. Symmetrically, pages *history.html*, *present.html* and *staff.html* are only in English, but are referenced also by Italian pages.
The unintended language switch present in IEN is particularly awkward: if *Pagina4.html* – an Italian-only page – is reached from an Italian page and then the standard time is accessed, the English version of the related page is showed (*stittime.shtml*), although the Italian version does exist (*stittime_i.shtml*).

AMAZON
The well-known e-commerce site AMAZON for the purchase of books, music and other goods was tested using our tool in 2001. A strange behaviour, which we have classified as a potential improvement area, was revealed by one of the test cases produced by **TestWeb**. The anomalous behaviour is associated with the operations: *change quantities of books purchased* and *add wrapping/message*. Let us suppose to be in the following situation. One book is in the shopping-cart and we want to carry out the following steps: (1) changing the number of books purchased, from 1 to 2, and (2) adding a wrapping. If we carry out these two operations in sequence, the result is 2 packages containing a book each and 2 separate messages. If we invert the operations, adding the wrapping before changing the quantity, the result is different. We obtain only one package containing 2 books and 1 message. Thus, the operations *change-quantity* and *add-wrapping/message* are not commutative (in the 2001 version of AMAZON).

NATURALIA

NATURALIA, a medium Web site (715 pages) that promotes national parks in Europe, was analysed in 2004 using **ReWeb**. The application of our clustering method revealed a potential intervention of restructuring. The grouping of the pages proposed by our algorithm was totally different from the actual decomposition into directories of the site. In this Web site, directories are structured by nation, while the clustering algorithm based on the page keywords grouped together pages with a similar content, irrespective of the related park nation.
Another problem of this site was discovered by means of clone analysis. The result is that 16 pages were clones. An example of cloned pages is shown in Figure 7. In such a case, a better solution would be a unique dynamic page that extracts the nation-related information from a database and generates the HTML pages to be displayed in the browser dynamically.


italia.html - Mozilla
http://www.naturalia.org/PARCHI/italia/italia.htm
ITALIA
Generalità
Parchi e riserve naturali
Aree protette
Home
francia.html - Mozilla
http://www.naturalia.org/PARCHI/francia/francia.htm
FRANCIA

Dati

- Superficie: 301.302
- km²Popolazione: 56.778.000
- Percentuale aree protette: 6,63%

Specie minacciate

- mammiferi: 3
- uccelli: 19
- piante: 151

Principali problemi ambientali
Inquinamento dell'aria in particolare nei centri urbani ed industriali.
Deterioramento della qualità delle acque dei fiumi e di falda.
Elevata produzione di rifiuti industriali ed urbani.

Dati

- Superficie: 543.998
- km² Popolazione: 56.138.000
- Percentuale aree protette 8,7%

Specie minacciata

- mammiferi: 6
- uccelli: 21
- piante:112

Principali problemi ambientali

inquinamento delle acque dei fiumi dovuto alla elevata presenza di nitrati nell'acqua provenienti dall'agricoltura;
inquinamento dell'aria nelle grandi aree urbane;
danni alle foreste dovute alle precipitazioni acide (circa il 20% degli alberi soffre di questi problemi).

Figure 7: *Example of cloned Web pages.*

AVAILABLE TOOLS

A lot of tools evaluating and improving quality aspects of Web applications [8, 9, 10, 13, 14] have been presented by researchers and industry.
Spiders (also called *crawlers* or *robots*) are software programs able to visit and download all the pages of a Web site. They usually follow the hyperlinks in each downloaded page, until all the reachable pages are obtained. **Webbot**[1] is a spider with support for regular expressions, used to specify which links are followed and which are excluded during the traversal. **WebSPHINX**[2] is a Java class library and interactive development environment for Web spiders. It provides support for writing customizable Web spiders in Java. An interesting feature of **WebSPHINX** is the graphical user interface, which can visualize the recovered Web pages and links as a graph. The main

[1] http://www.w3.org/Robot/
[2] http://www-2.cs.cmu.edu/~rcm/websphinx/

difference between the **ReWeb** spider and these spiders is that **ReWeb** is able to manage dynamic pages while the others are not so.
SmartSite [12] is a tool oriented toward Web site evolution and offering advanced reverse engineering capabilities. This tool is useful to understand the overall site's structure. It provides many different types of views. **WARE** [13] is another tool for the reverse engineering of Web applications. The output of **WARE** consists of a set of UML diagrams, showing several aspects of a Web application at different abstraction levels. The recovered diagrams ease the comprehension of the application and support its evolution. **WARE** and **ReWeb** aim at the same kind of result: help the user understand and improve Web applications. However, the analyses and views implemented in the two tools are completely different. In particular, the diagrams extracted by **WARE** are more abstract (use cases, UML class diagram, etc.) than those available with **ReWeb** (structural view, system view).
An example of link checker is **Link Valet**[3]. Given the URL of a page, **Link Valet** fetches the page and prints a report including a summary of the status of all links contained in the page. The HTTP Status code of the URL is highlighted to indicate a recent update or an error condition (e. g., *404 Error - File Not Found*). The analysis of the "pending links" implemented in **ReWeb** is able to find "ghost pages", which correspond to **Link Valet**'s "pending links". However, the report produced by **Link Valet** is more detailed.
The Web site of the W3C[4] gives information about evaluation, repair and transformation tools that make the Web more accessible. Evaluation tools perform a static analysis of the Web pages, focused on their accessibility, and return a report or a rating. Repair tools assist the author (through semi-automatic transformations) in making the pages more accessible.
Bobby and **Doctor HTML** are two evaluation tools. **Bobby** helps authors determine if their sites are accessible for users who have some form of disability and it also analyzes the Web pages for compatibility with various browsers. **Doctor HTML** performs minimal accessibility checking, verifies the links, checks the spelling and performs some syntax controls.
The **W3C HTML validation service** checks HTML documents for compliance with the W3C HTML Recommendations and other HTML standards. **Demoronizer** and **Tidy** are two repair tools. **Demoronizer** removes vendor specific HTML conventions and extensions, while **Tidy** repairs HTML errors (e. g., missing or mismatched end tags), improves the style and converts HTML to XHTML.
ReWeb does not check accessibility problems as **Bobby** and **Doctor HTML** and it does not repair HTML errors as **Demoronizer** and **Tidy.** Among the analyses implemented in

[3] http://www.htmlhelp.com/tools/valet/
[4] *http://www.w3.org/WAI/ER/existingtools.html*

ReWeb, two are completely new, “Multilingual Web Site Consistency” and “Keyword Based Clustering”.

Even if usability [11] is another important concern for Web applications, this kind of analysis is not considered in **ReWeb** and not treated explicitly in this paper. A novel approach for automatically evaluating the usability and accessibility of Web sites performing a static analysis of their Web pages was presented by Vanderdonckt and Beirekdar [9]. A survey of automatic usability evaluation tools for Web sites was proposed by Brajnik [8].

Recently, a few testing tools have been proposed to support functional (black box) testing of Web applications. These black-box testing tools are based on capture/replay facilities: they record the interactions that a user has with the Web application and repeat them during regression testing. One of them is **WEBART**[5].

HttpUnit[6] (an extension of **Junit**) is a suite of Java classes to test Web applications. **HttpUnit** emulates the way a Web browser works. It makes a page request and it parses the response. It is able to black box test entire Web applications and it manages cookies. If, for example, the Web application includes a shopping cart, a test can be written to: log-in, select an item, place it in the shopping cart and check-out. **WEBART** and **HttpUnit** are black-box testing tools, while **TestWeb** is a white-box testing tool. The idea of using the model extracted from a Web application for the generation of test cases has been introduced for the first time with **TestWeb**.

Elbaum et al. [14] have proposed a Web application testing tool implementing structural testing and utilizing the data captured in user sessions (stored in a modified log file) to create the test cases automatically. Instead of inserting manually the inputs for the test cases, as done with **TestWeb**, the inputs are recovered automatically from the modified log file. The coverage is still guaranteed and the process is completely automatic. This approach can be considered an improvement of **TestWeb**.

For each tool explicitly cited in this section, Table 2 summarizes its category, availability (research prototype, free, commercial) and anomalies/failures it can detect.

Tool	Category	Availability	Anomalies/Failures
Webbot	Spider	freely available	navigation problems
Websphinx	framework for spider construction	freely available	navigation problems and (partially) defects in high level design
Smartsite	reverse engineering	commercial	navigation problems, (partially) defects in high level design, markup errors, non-compliances with standards.
WARE	reverse engineering	research prototype	navigation problems, defects in high level design and cloned pages
Link Valet	link checker	freely available	navigation problems
Bobby	accessibility checker	commercial	accessibility problems
Doctor HTML	Web page analysis	commercial	accessibility problems, navigation

[5] http://www.webtesttools.com/webtesttools/score.asp?FileID=62713

[6] http://httpunit.sourceforge.net/

	tool		problems, spelling problems and markup errors
HTML Validation	standards validator	freely available	non-compliances with standards
Demoronizer	HTML cleaner	freely available	non-compliances with standards
Tidy	cleaner and XML transformer	freely available	non-compliances with standards, accessibility problems and markup errors
WebArt	link checker and testing tool (capture/replay)	commercial	navigation problems and deviation from the expected behaviour
HTTPUnit	testing framework (black box testing)	freely available	deviation from the expected behaviour
Elbaum et al.	testing tool (structural testing)	research prototype	deviation from the expected behaviour

Table 2: *Analysis and testing tools for Web applications.*

CONCLUSIONS

Some general considerations can be made, out of the empirical work conducted in the last few years and summarized above. The fact that the quality of existing Web applications is in general poor seems to be not only a common idea of the practitioners and of the Internet users, but it was empirically detected in the field. Our critical review [5] showed that only 40% of the Web applications considered in our study exhibit no anomaly and failure.
Moreover, our study indicates that evaluation tools such as **ReWeb** and **TestWeb** are effective in discovering such anomalies and failures. However, the most advanced features (e.g., reverse engineering of high level models, structure-based testing) are offered only by research prototypes. Thus, for a wide adoption of such techniques, their integration into widely used development environments is advisable. For example, no commercial or freely available tool supports the detection of multilingual problems and only research prototypes support test case generation.

REFERENCES

[1] Ginige, A., & Murugesan, S. *Web engineering: An introduction*. IEEE Multimedia, 8(1):14-18, April-June 2001.
[2] Ricca F., & Tonella P. *Understanding and Restructuring Web Sites with ReWeb*. IEEE Multimedia, 8(2):40-51, April-June 2001.
[3] Ricca F., & Tonella, P. *Analysis and Testing of Web Applications*. ICSE 2001, International Conference on Software Engineering, Toronto, Ontario, Canada, May 12-19, pages (25-34), 2001.

[4] Baxter I.D., Yahin A., Moura L., Sant'Anna M., & Bier L. *Clone Detection Using Abstract Syntax Trees*. ICSM 1998, International Conference on Software Maintenance, Bethesda, Maryland, USA, November, pages (368-377), 1998.

[5] Ricca F. & Tonella P. *Anomaly Detection in Web Applications: A Review of Already Conducted Case Studies* (awarded as the best paper of the conference), CSMR'2005, European Conference on Software Maintenance and Reengineering, The Manchester Conference Centre, Manchester, UK March 21-23, 2005.

[6] W3C Open Source Software. *http://www.w3.org/Status.html.*

[7] Offut J. *Quality Attributes of Web Software Applications.* IEEE Software, 19(2):25-32, March-April 2002.

[8] Brajnik, G. *Automatic web usability evaluation: what needs to be done?*, Human Factors and the Web, 6th Conference, Austin, June 2000, *www.dimi.uniud.it/~giorgio/papers/hfweb00.html*.

[9] Vanderdonckt J., Beirekdar A. *Automated Web Evaluation by Guideline Review*. Journal of Web Engineering 4(2): 102-117 (2005).

[10] Fraternali P., Lanzi P.L., Matera M., Maurino A. *Model-Driven Web Usage Analysis for the Evaluation of Web Application Quality*. Journal of Web Engineering, Vol. 3, No. 2 (2004), pp. 124-152, Rinton Press.

[11] Ivory M., Hearst M. *The state of the art in automating usability evaluation of user interfaces*. ACM Computing Surveys 33(4): 470-516, 2001.

[12] Tilley S, Huang S. *Evaluating the Reverse Engineering Capabilities of Web Tools for Understanding Site Content and Structure: A Case Study*. ICSE 2001, International Conference on Software Engineering, Toronto, Ontario, Canada, May 12-19, pages (514-523), 2001.

[13] Di Lucca G.A, Fasolino A.R., Pace F., Tramontana P, de Carlini U. *WARE: A Tool for the Reverse Engineering of Web Applications*. Conference on Software Maintenance and Reengineering 2002, Pages: 241 – 250.

[14] S.Elbaum, S.Karre, G.Rothermel. *Improving Web Application Testing with User Session Data*. ICSE 2003, International Conference on Software Engineering, Pages: 49 - 58

----*sidebar*---- Technical characteristics of ReWeb and TestWeb----------

The two tools **ReWeb** and **TestWeb** have been developed to support analysis and testing of Web applications. Both tools perform their operations on an abstraction of the Web applications, indicated in Figure 1 as UML model. The most important entities of a Web application as, for example, static and dynamic pages, links, frames and forms are explicitly represented in the model [T1].

The **ReWeb** tool [T1] consists of three modules: a Spider, an Analyzer and a Viewer.

- The Spider downloads all pages of a target web site, starting from a given URL and providing the input required by dynamic pages, and it builds a model of the downloaded site (see [T1] for the pseudo-code of the spider). Each page found within the site host is downloaded. The HTML documents outside the web site host are not considered. Dynamic pages pose some problems to the activity of the Spider. Since the content of these pages is decided at run time, it may in general depend on the input previously provided by the user. In particular, the structure of a dynamic page may change when it is encountered in a different interaction. Since the model of a Web site encompasses all possibilities, the Spider has to recover all variants of a dynamic page and has to merge the equivalent ones into a single representative object. This can be achieved by specifying the input values to be provided before downloading the dynamic pages of interest. Moreover, the same dynamic page is downloaded several times, with different inputs, when the different conditions generate a different page structure. All sequences of input values to be provided before each page download are specified in a file which is read by the Spider. All dynamic pages specified in the file are downloaded after providing the Web server with the given inputs. Finally, all equivalent versions of the same dynamic page are merged [T1].
- The Analyzer uses the UML model of the web site and the downloaded pages to perform several analyses. Since the structure of a Web application can be modelled with a graph (basically, a re-interpretation of the UML model), several known analyses, working on graphs, such as flow analysis [T2] and traversal algorithms can be applied. An example of analysis which specialize the framework of flow analysis is the computation of the **reaching frames**, which determines the set of frames in which each page can appear. **Multilingual Web site consistency** [T3] is checked using two phases that correspond to two algorithms. The *language identification* phase, which analyzes the content of the pages in the site and partition them according to the language (the algorithm uses the open source software *Textcat*) and the *page matching* phase. The result of the *page matching* algorithm [T3] is a set of corresponding pages in different languages (pages P and Q are corresponding if the edit distance between the syntax trees resulting from parsing the HTML source code for P and Q is the

minimum among all pairs of Web pages in the two languages; content translation is accounted for in the computation of the distance) as well as a set of pages with no counterpart (see Figure 6). **Multilingual hyperlink analysis** is executed using an algorithm, working on the model, which checks the symmetry between corresponding hyperlinks. For example, if a hyperlink connects a page p_it to a page q_it, and q_eng and p_eng are the corresponding English pages, a symmetric hyperlink from p_eng to q_eng is expected to exist. **Clone analysis** uses the *structural edit distance* to discover pages that have a similar structure while **Keyword based clustering** [T4] employs Natural Language Processing techniques to extract keywords from Web pages. Pages that share a large number of keywords are grouped together by an agglomerative hierarchical Clustering algorithm.

- The Viewer provides a Graphical User Interface (GUI) to display the Web application views (*structural* and *system*) as well as the textual output (reports) of the analyses. The graphical interface used to display views supports a rich set of navigation and query facilities. Among the provided facilities, the Viewer supports zoom, search, deletion of incoming or outgoing edges, and focus. The facilities for focusing on and searching a node are useful when the visualized graphs are very large. By exploiting the focusing facility it is possible to display only a limited neighbourhood of a selected node.

Web Spider and Analyzer are written in the Java language, while the Viewer is based on Dotty (Dotty is a customizable graph Editor developed at AT&T Bell Laboratories by Eleftherios Koutsofios and Stephen C. North).

TestWeb [T1] consists of two modules: the *Test generator* and the *Test Executor*. The *Test generator* is able to generate test cases from the UML model of a Web application. The user has to add some information to the model produced by **ReWeb** to complete it for testing purposes and furthermore the user has to choose a test criterion. The Test generator computes the path expression (algebraic representation of all paths in a graph) of the model [T1] and uses it to generate sequences of test cases which satisfy the coverage criteria chosen by the user. The user specifies the page type when the distinction between static and dynamic pages cannot be obtained automatically. The user also provides the set of used variables for each dynamic page whose content depends on some input value. Finally, the user has to attach conditions to the edges whose existence depends on the input values. Input values in each URL sequence are left empty by the Test generator, and the user has to fill them in, possibly exploiting the techniques traditionally used in black box testing (boundary values, etc.).

TestWeb's *Test executor* can now provide the URL request sequence of each test case to the Web server, attaching proper inputs to each form.

REFERENCES

[T1] Ricca F. & Tonella P. *Building a Tool for the Analysis and Testing of Web Applications: Problems and Solutions.* TACAS 2001, Tools and Algorithms for the Construction and Analysis of Systems, Genova, Italy.

[T2] Aho A.V., Sethi R. & Ullman J. D. *Compilers. Principles, Techniques, and Tools.* Addison-Wesley Publishing Company 1985 Reading, MA.

[T3] Tonella P., Ricca F. , Pianta E. & Girardi C. Restructuring Multilingual Web Sites. ICSM 2002 International Conference on Software Maintenance.

[T4] Tonella P., Ricca F., Pianta E. and Girardi C. Using Keyword Extraction for Web Site Clustering. WSE 2003, International Workshop on Web Site Evolution, Amsterdam.

--