Rule-based structural analysis of web pages

Fabio Vitali, Angelo Di Iorio, Elisa Ventura Campori

Department of Computer Science
University of Bologna
fabio@cs.unibo.it

Abstract. Structural analysis of web pages has been proposed several times and for a number of reasons and purposes, such as the re-flowing of standard web pages to fit a smaller PDA screen. elISA is a rule-based system for the analysis of regularities and structures within web pages that is used for a fairly different task, the determination of editable text blocks within standard web pages, as needed by the IsaWiki collaborative editing environment. The elISA analysis engine is implemented as a XSLT meta-stylesheet that applied to a rule set generates an XSLT stylesheet that, in turn, applied to the original HTML document generates the requested analysis.

1 Introduction

The HTML language was born to provide a simple and easy-to-learn markup language for scientists to write their papers and publish them on the Internet. With time, most markup of a web page has come to refer to graphics and layout, and the actual content of the page is now often heavily intermixed to structure and graphics, and it is rather hard to extract and analyze.

Applications aimed at providing different functionalities on web content than visualization have therefore a hard task in extracting relevant information from web pages. Sophisticated searching and adaptive content delivery are just two examples where the actual HTML code provided by the web server needs to be decomposed, analyzed and recomposed in a different manner for the application to deliver its functionalities.

In this paper we introduce a new kind of application where the structural analysis of HTML documents is necessary: the in-place editing of web pages for customization of content. Personal variants of web pages have been at the basis of the original vision for distributed hypertext systems, as described by Ted Nelson for his Xanadu environment [14]. Wikis [7] are recent tools that re-introduce universal editing of web pages and reduce considerably the complexity of the task of editing web content. On the other hand the presentation complexity of wiki pages is rather meager, and the application only allows editing of pages belonging to the wiki itself.

The IsaWiki system [8] aims at providing easy tools for accessing and editing pages belonging to any web site. By allowing in-place editing of any web page, and by storing the edited content in the wiki site, ready for delivery any time the same user accesses the same page again, IsaWiki aims at creating an environment for personal customization of web content much beyond the reach of wiki systems.

One of the issues discussed in the design of the IsaWiki system was whether the user would be interested in editing any part of the web page, or whether we should provide some tools to identify those parts of the HTML code that contained the actual content, and provide editing facilities just for them. Having discarded the idea of just providing a complete access to the whole of the web page (the layout and overall structure of a professionally designed HTML page is often rather complex, difficult to understand even to expert web designers), we decided to consider a web page as a layout of independent substructures, and to provide editing access to just those substructures that contain the actual and specific content of the page.

For this reason we implemented a client-side tool, called elISA (Extraction of Layout Information via Structural Analysis), that is aimed at providing a complete structural analysis of the web pages, in order to determine their content areas and activate the editing facilities just on them.

The elISA system is a rule-based application that seeks patterns in the HTML code and labels structures and substructures of the HTML page according to these patterns. It is actually implemented as a transformation engine that, once applied to HTML pages, adds appropriate attributes and other markup to the original document. The editing part of the IsaWiki system will then search those attributes and activate the editing facilities just on those parts that are marked as content areas.

Of course these rules are far from perfect and universal. They are based on an analysis of the most frequent cases of HTML structures as described in the literature and available on the web, so they certainly do not cover all situations. Furthermore best practice evolve with time, so it may well be that new types of pages (with different applicable rules) become fashionable on the web in the future. For this reason, elISA clearly differentiates between the rule engine and the actual rules, which are declaratively expressed using an appropriately defined XML-based language. New fashions in web page constructions, or new and more precise rules for existing structure, then, will require no intervention in the fundamental elISA engine, but just the specification of one or more additional declarative rules.

In this paper we wish to describe the elISA analysis tool, and show how it fits in the overall IsaWiki application. In section 2 we plan to discuss a few related works in the area of web page analysis, and describe the kind of rules they apply in their activities. In section 3 we describe IsaWiki and its overall architecture, providing a justification for the creation of the elISA system. Section 4 describes the overall architecture of elISA, including the basic overall transformation mechanism (based on XSLT meta-stylesheets) and the syntax for expressing analysis rules. Section 5 describes the actual rules we are now using in our analysis of web pages, and that are in many cases drawn from the systems described in section 2 and re-expressed in the syntax described in section 4.

2 Structural analysis of web pages

The HTML language was born to provide a simple and easy-to-learn markup language for physicists to write their papers and publish them on the Internet. The language provided initially just a few structural and typographical constructs as well as the new and relevant concept of the hypertext link.

The advent of the Mosaic browser paved the road for the World Wide Web to become the main tool for the information highways. Soon, every individual, company and organization had to have their own web site, and just could not accept the idea of settling for the kind of simple pages that HTML, properly used, allowed them. By means of HTML tricks, tag exploitation, and creative use of graphics, page designers have managed to make a boring structural markup language become the means for incredibly complex and sophisticated interactive events available through web browsers.

Of course, the drawback of this evolution was that most professionally created web pages started to include large markup sections aimed at decorative and layout purposes, and that these are often intermingled with the actual content of the page: although human readers can in most cases easily tell apart content and presentation, machine interpretation of the content is seriously hindered.

Not even XML has provided a solution for this situation: although XML has been a huge success and a technology and implemented by almost every player in the software industry, its main use nowadays is behind the scenes, server-side, while what gets delivered to the browser is still just an HTML document generated on the fly.

Still, applications that require to identify and classify subparts of web pages is high, and their aims are as wide and numerous as the applications themselves, but surely include content repurposing for special devices (e.g. mobile computers), data mining, and summarization. Even the techniques they implement vary considerably.

For instance in [11] a hierarchical representation of the screen coordinates of each page element is used to determine the most common areas in the page, such as header, side menu (either left or right), main content area, and footer. This analysis exploits the expected structural similarities between professionally designed pages as suggested by usability manuals and implemented by competitors.

[20], [13] and [15] all propose a semantic analysis of the structure of the HTML page, aiming at the discovery of visual similarities in the contained objects in analogous pages. The fundamental observation is that the standardization in the generation tools of web pages has created consistencies in the style of headings, records and text blocks of the same category. Unfortunately, there are many ways in HTML to obtain the same effect in terms of font, color and style (tag names, order of tag, use of inline, internal or external CSS styles, etc.). So clearly similarities in the final effect may well correspond to differences in the underlying code, which adds a further layer of complexity in the process. Also [21] propose a method for classification of elements in a web page based on their presentation, nature and richness in style information.

In [5] the repurposing of web content for PDA devices is obtained by determining the criteria for identify the elements of the page that constitute a content unit. The process is iterative, starting from a single block as wide as the whole page, and then progressively determining sub-areas within the areas already determined, while in [9] the reformatting of the web content for smaller PDA screens is performed through the filtering of specific nodes of the DOM tree and leaving only relevant nodes. Another contribution to the reformatting of web pages for small screen is [16]

3 IsaWiki: a collaborative editing environment

The World Wide Web is a powerful means to publish information and provide services but it lacks several hypertext functionalities [4]. In particular, there is still a strong difference between the roles of the author and the reader: readers use free and easy tools but can only choose reading paths explicitly provided for by the authors and cannot create new content, links or personal variants of web pages; on the other hand authors need to master a large number of different technologies, and/or to use complex and expensive tools. Although web pages can be created with simple tools (such as by saving text content as HTML in Microsoft Word), the result is certainly not satisfying, and good looking web pages still require the intervention of technical-savvy, competent professionals. Furthermore, all authors can only edit a restricted set of resources (the ones on which they have write permission).

It is possible to realize a "writable Web" with current web technologies? The IsaWiki project set out to prove that these functionalities can be provided without revolutionizing the current architecture of the World Wide Web or introducing new standards and protocols [18].

Two solutions are recently gaining importance in the collaborative editing panorama: wikis and weblogs. Weblogs [3] are tools for editing (mostly based on web forms) and publishing of personal diaries addressed to individuals and small communities. Although weblogs allow users to easily and fast edit web pages during the browsing activities, the collaboration framework they provide is incomplete: a weblog page is a sequence of posts ordered by date and time, rather than a page composed by personal interventions by different users, in different times, on different fragments of the document. More advanced functionalities are provided by wikis [7], collaborative tools for shared writing and browsing, allowing every reader to access and edit any page of the site, through simple web forms and a very intuitive text based syntax for special typographical effects. Wikis, too, show some relevant drawbacks towards reaching a full writable web: graphical sophistication is still too raw, a new syntax needs to be mastered and, above all, only pages already stored on the wiki-site can be edited.

Indeed, in a universal editing environment personal interventions need to involve every web page on every web server, regardless of users' writing permissions. Many researchers and professionals have proposed on-the-fly addition of contents and links to existent documents towards this goal: whenever a user accesses to a page and a customized version of this page does exist, the application retrieves the original document from the origin server (which remains unmodified) and enriches it with information stored in external databases.

For instance, CritLink [19] was the first annotation system for the World Wide Web based on a proxy, followed by a lot of similar free or commercial projects such as Commentor [6], iMarkup [10], or the now-defunct eQuill and Thirdvoice. Recently the W3C has proposed a shared Web annotation system (and an underlying protocol to make it work), Annotea [12] based on the same schema: Annotea clients, such as Amaya [1] or Annozilla [2], request the annotations (expressed in RDF syntax) from pre-established Annotea servers and show them side by side with the original unmodified documents.

It is important to notice that annotations systems provide only a partial support for customization: a document, in fact, is only composed of two overlapping layers, the original document and the annotations layer that is attached over the main one. Thus readers cannot delete contents or re-organize the document's structure, and furthermore multiple versions management and backtracking cannot be created.

These solutions meet only some of the requirements we believe are necessary in a full writable web environment. IsaWiki [8] is a web editing environment being developed at the University of Bologna. IsaWiki allows every user to easily create, modify and above all customize web pages while browsing. Any web page can be edited even if no write permission on the resource has been granted.

IsaWiki users browse web pages normally through their browser (currently Internet Explorer only, but implementation for Mozilla is well under way). By activating the IsaWiki service, a sidebar would appear during the navigation. Whenever the requested URL changes and the actual document is downloaded from its origin server, the IsaWiki client would interrogate the IsaWiki server asking for modifications to the requested document. If a variant of the accessed page is present, the server would send it and it would be displayed in the browser instead of the original page. Note that the original resource remains unmodified on the origin server.

By selecting the edit command in the IsaWiki sidebar, a content editor appears and allows the user to add, delete and modify the content of the page. Upon saving the changes, the modifications are sent to the server and made available to all further subscribers. Closing the editing session, the navigation behaves normally. Multiple versions of the same page can be kept and displayed individually, by selecting them in a version list always displayed in the IsaWiki sidebar.

Editing is performed in place, directly within the web page and the browser being used. Recent browsers allow users to edit content through a few recently introduced properties of the page elements, such as *contentEditable* in Internet Explorer and *designMode* in Mozilla. This allows inline editing of the content within the actual page that hosts the original content, with the same styles and layouts. These modifications affect the local copy of the document that need to be posted on a server (in our case, the IsaWiki server). Figure 1 shows different editable blocks in a new IsaWiki document:

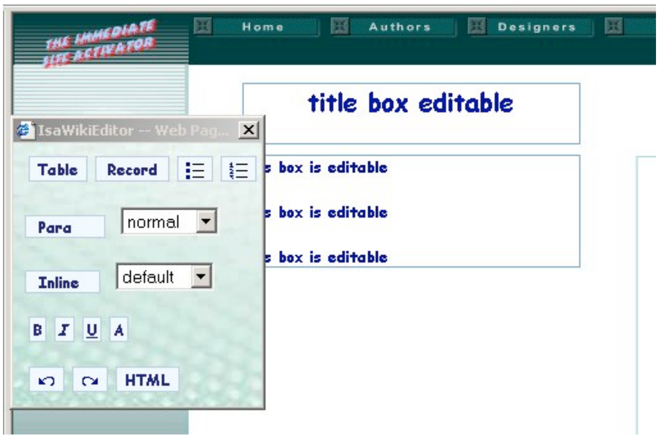

fig. 1: Some editable regions in a IsaWiki document

An important aspect of IsaWiki is deciding which parts of the web page are to become editable: rather that allowing editing on the whole web page, in fact, IsaWiki tries to identify editable regions (content areas) and non-editable regions, usually empty or decorative o navigational areas.

To perform this task, IsaWiki looks for regularities in the HTML code of the page. In fact, template-driven automatic page generation is often used by both HTML editors and content management systems (CMS). Content and layout are separately stored on the CMS, and, whenever a user asks for a resource, the server automatically merges them into a complete web page before delivery. For instance, IsaWiki actually started off a simple web-page creation tool called ISA [17], a system to easily create sophisticated web sites exploiting well-known interfaces such as MS Word for content creation (having no knowledge of HTML or any other markup language) and Photoshop or other graphical tools for layout (without having great expertise but simply drawing the layout and slicing it in different zones).

IsaWiki knows all about ISA regions, and can determine their presence and role in the final web pages. Unfortunately, this is not true for all template-driven web pages, where the distinction between content and layout need to be deduced by careful examination of the HTML code of the page. For this reason we have built the elISA engine. Started off as a simple function within the client-side editor, elISA has evolved to become a full-featured structural analysis tool for web pages, amenable to being applied in many different tasks. Within IsaWiki, it is used to determine what are the main areas in the web page, and to determine which of them are content areas, that are worth to be modified by the user through the IsaWiki editor.

4 The elISA architecture and language

The elISA engine (elISA stands for *Extraction of Layout Information via Structural Analysis*) is an independent module of IsaWiki in charge of identifying content regions of a web page through an heuristic approach. The elISA engine reads the document loaded by the browser and outputs a list of editable fragments that the IsaWiki application transforms in *contentEditable* or *designMode* elements in the DOM of the web page. elISA determines these parts according to a number of typical layouts or layout details that are used consistently by page designers throughout the web: for example the positioning of the links, the organization of the frames, the location of the banners and so on. To do so, it relies on declarative rules that are expressed with a specific rule language, and then applied to the web page via a meta-stylesheet in XSLT. In section 5 we will expose and discuss some of these rules. Here we will describe the actual working of the elISA engine itself.

The distinction between the analysis engine and the actual rules is important: whichever rules we would end up choosing, and however detailed and precise our implementation of the rules would turn out, we were sure that in a few years, months possibly, new web styles and new coding techniques would come out that made our rules obsolete or of only limited usefulness. Thus we decided to create a generic evaluation engine that is fed with fresh declarative rules every time, so as to make it simple to keep the rules up-to-date.

Furthermore, even if elISA is primarily used for the determination of editable text areas within the IsaWiki editor, its rule-based approach allows for more and more detailed analysis of any kind of regularity in web pages.

The execution cycle of the elISA engine within IsaWiki is composed of three steps:

- **Cleansing of the HTML code** (reduction to well-formedness): since the actual evaluation of the elISA rules is performed through an XSLT meta-stylesheet, it is necessary to create an equivalent version of the HTML document that is at least well-formed. The elISA parser works within IsaWiki, which is a sidebar within the browser, so that it is possible to make use of the HTML parser of the browser itself.
- **Evaluation and applications of the rules on the HTML code**: The elISA engines then loads the rules file, as written in the elISA rule language, and applies an XSLT stylesheet to it. The end result is an XSLT stylesheet that, applied to the source document, creates a list of elements that have a clearly identified semantic role (such as logo, footer, navigation bar, advertisement, content area, etc.). The syntax of elISA rules is discussed in the following.
- **Application of the editable attributes to the relevant HTML fragments of the code**. Once the list of content areas is determined, they are found again in the original document and are set as editable before the document is actually made available to the user for editing.

Obviously, only the second step of the sequence does any actual analysis of the page, and constitutes the core of the elISA engine. The other steps are necessary for the blending of the elISA engine within the IsaWiki application.

At the basis of the elISA engine is a rule language for expressing regularities in the source HTML document. Fig 2 shows an example of the rule:

```
<RULES identity=”false”>
    <RULE context=”TH|TD”>
          <CALL name="nodeTextExceptTable">
            <ADD select=".//text()"/>
            <EXCEPT select=".//TABLE//text()"/>
            <EXCEPT select=".//FORM//text()"/>
            <EXCEPT select=".//OPTION//text()"/>
            <EXCEPT select=".//TEXTAREA//text()"/>
            <EXCEPT select=".//LABEL//text()"/>
            <EXCEPT select=".//BUTTON//text()"/>
          </CALL>
          <CHECK>
            <WHENEVER test="count($nodeTextExceptTable)=0">
              <SET attr="bgcolor">orange</SET>
              <SET attr="elISAtype">layout</SET>
            </WHENEVER>
          </CHECK>
    </RULE>
    <RULE>
          ...
    </RULE>
    ...
</RULES>
```

fig. 2: a simple rule for checking empty layout cells

The rule is applied in the context of `TD` and `TH` elements, i.e., whenever the elISA engine examines one of these tags in the HTML document. The rule is composed of the definition of the variable “`nodeTextExceptTable`”, which contains a node set of all non-empty text nodes within table headers and table cells, and the rule that checks if the number of nodes in the variable is greater than 0. If this is the case, it sets two attributes to the context element: a background color and an `elISAtype` attribute set to “layout”. The actual meaning of this rule is to identify empty layout cells as table elements that have no text content, but may have form elements or other nested tables inside.

All rules are inserted in `<RULE>` elements, which define their context, i.e., the scope within which the rule has to be applied. Each context is specified by an XPath, which could be as simple as the name of an element or as complex as a full and complex tree-traversal specification. Within the `<RULE>` element there are as many `<WHENEVER>` elements as checks needing to be performed, each specifying by means of a boolean XPath a condition to check on the context elements. `<WHENEVER>` elements can nest, to perform additional subchecks, and can also contain specifications for the modification of the source HTML document. In this case, two new attributes are set to the context elements, but it is also possible to modify existing attributes, or to output new content within the context.

Variables are useful to store and access with a simple name the node sets whose determination may require long and convoluted XPath expressions and that are used frequently in the rule. `<CALL>` elements are used to define new variables, which may appear both globally (outside of `<RULE>` elements) and locally (inside of `<RULE>` elements). Since quite often the formula to select the nodes of the variable are complex and lengthy, and are composed of a positive formula and one or more exceptions, the `<CALL>` element contains two subelements, `<ADD>` and `<EXCEPT>`, to specify positive formulas and exceptions separately. Each, of course, requires an XPath expression.

The elISA rule language also contains a few additional functions that can be used in the XPath formulas, such as `nchar($var)`, which returns the number of characters in all the strings contained in `$var`, or `lc($var)` and `uc($var)` that convert the text in `$var` in lowercase (respectively uppercase), or `differentChar($var,$char)`, that is true if in `$var` there is at least one character not contained in `$char`. The use of some of these functions will be shown in section 5.

```
<xsl:template match=”TD|TH”>
          <xsl:variable name="nodeTextExceptTable"
   select=".//text() and not((.//TABLE//text()) or
   (.//FORM//text()) or (.//OPTION//text()) or
   (.//TEXTAREA//text()) or (.//LABEL//text()) or
   (.//BUTTON//text())"/>
          <xsl:copy>
           <xsl:if test="count($nodeTextExceptTable)=0">
             <xsl:attribute
name=”bgcolor”>orange</xsl:attribute>
             <xsl:attribute
name=”elISAtype”>layout</xsl:attribute>
           </xsl:if>
           <xsl:apply-templates/>
          </xsl:copy>
</xsl:template>
```

fig. 3: the XSLT template corresponding to the elISA rule in fig. 2

The simplicity of the elISA language and the systematic use of XPath expressions lead us to implement the analysis as a transformation of the source document according to an XSLT stylesheet. In fact each `<RULE>` of the document can be transformed rather easily into an XSLT template (see fig. 3 for the corresponding template).
Thus the elISA engine is simply a converter that creates an XSLT stylesheet out of the rule set, and then applies it to the source document. The conversion is performed through a meta-stylesheet that creates XSLT templates out of the elISA rules.
The final output of the transformation is composed only of those elements that matched one of the rule contexts, thus making a list of elements with an elISA attribute specifying their role. But elISA can be used to output the very same input document with added attributes and content. The root element of the elISA rule language is the `<RULES>` element, which has an `identity` attribute. When `identity` is false, only the matching elements are put out, thereby creating a simple list of matching elements. When `identity` is true, on the other hand, an identity template is activated, and every single node (element, attribute, text, comment, etc.) of the input document is put in the output document, with the additional attributes specified in the actual rules, of course.
In the example shown in fig. 2 and 3, besides setting the `elISAtype` attribute that is used by the IsaWiki application, the elISA rule also sets a color for the background of the cell. This attribute is specified for demonstration purposes only, to verify that the elISA application is working correctly. To debug the rule set, in fact, we also specify some kind of visual effect (that would disappear in the final rule set) for elements we are setting an `elISAtype` value on. Some example of colored pages are given in the web site accompanying the conference.

5 Determining editable regions in web pages with elISA

In this section we first summarize a few important rules that we are actually applying to web pages in the context of IsaWiki, in order to determine the content areas to be made editable by the IsaWiki editor, and then we provide a few intermediate experimental results on how well these rules are performing for the set of pages we have chosen for our tests.

5.1 Rules for textual content detection

Although the whole rule set is rather long and convoluted, we show here only the rules we have created for table cells and headers (`context="TH|TD"`). In the examples we have also left the color coding in the rules for clarity.
The first and foremost check is the determination of empty layout cells. This is the `<WHENEVER>` element shown in fig. 2, and it will not be repeated here. Within the same `TH|TD` context, though, many additional `<WHENEVER>` checks are being performed. We put in the nodeText variable those text nodes that do not contain links, and in nodeLink all links that do not contain tables or form elements. See fig. 5 for the actual elISA code.

```
<CALL name="nodeText">
           <ADD select="$nodeTextExceptTable"/>
           <EXCEPT select=".//A[@href]//text()"/>
</CALL>
<CALL name="nodeLink">
           <ADD select=".//A[@href]//text()"/>
           <EXCEPT select=".//TABLE//text()"/>
           <EXCEPT select=".//FORM//text()"/>
           <EXCEPT select=".//OPTION//text()"/>
           <EXCEPT select=".//TEXTAREA//text()"/>
           <EXCEPT select=".//LABEL//text()"/>
           <EXCEPT select=".//BUTTON//text()"/>
</CALL>
```

fig. 5: two variable definitions for table cells

If there are both link elements and text elements, then we apply a rule described in [9] that requires computing the ratio between the number of links and the number of non-link words in the context. By number of non-link words we consider the number of non-link characters divided by the average length of word, which is set to five. The threshold for determining whether a table cell is a navigation element is set to 0.25 (actually in our experiments, as discussed in section 5.2, we have preferred to use 0.30 as the threshold value, for it is giving better results in our opinion). This means that whenever the number of links divided by the number of words in a context is greater than 0.25, the context is supposed to be a navigational element, otherwise a content area. All formulas and values are taken from [9]. This corresponds to the elISA fragment shown in fig. 6.

```
<WHENEVER test="count($nodeLink)>0 and count($nodeText)>0">
     <WHENEVER test=
      "count($nodeLink) div (nchar($nodeText) div 5) > 0.25">
         <SET attr="bgcolor">pink</SET>
         <SET attr="elISAtype">link</SET>
           </WHENEVER>
           <WHENEVER test=
      "count($nodeLink) div (nchar($nodeText) div 5) < 0.25">
         <SET attr="bgcolor">cyan</SET>
         <SET attr="elISAtype">text</SET>
    </WHENEVER>
</WHENEVER>
```

fig. 6: the rules for classifying navigation elements and text areas

If the entirety of the content of the context is a link element, then this is definitely a navigational element, as is marked appropriately. Note that the background color goes from pink (quite sure) in the rule in fig. 6 to red (really sure) in the fragment in fig. 7.

```
<WHENEVER test="count($nodeLink)>0 and count($nodeText)=0">
    <SET attr="bgcolor">red</SET>
    <SET attr="elISAtype">link</SET>
</WHENEVER>
```

fig. 7: the rule for classifying link-only elements

Finally, contexts without link elements can be either non-empty cells, in which case they are surely content areas (shown in green), or empty cells, in which case they are surely layout cells (shown in brown). This corresponds to the rule in fig. 8.

```
<WHENEVER test="count($nodeLink)=0 and count($nodeText)>0">
    <WHENEVER test="not(differentChar($nodeText,' '))">
        <SET attr="bgcolor">brown</SET>
        <SET attr="elISAtype">layout</SET>
    </WHENEVER>
    <WHENEVER test="differentChar($nodeText,' ')">
        <SET attr="bgcolor">green</SET>
        <SET attr="elISAtype">text</SET>
        <WHENEVER test="nchar($nodeText)< 20">
            <SET attr="bgcolor">yellow</SET>
            <SET attr="elISAtype">layout</SET>
        </WHENEVER>
    </WHENEVER>
</WHENEVER>
```

fig. 8: the rules for classifying text-only cells

5.2 Experimental results with elISA rules

As mentioned, the elISA application can be used within the IsaWiki application for client-side web editing, as well as an independent and autonomous tool. The complete list of rules we are working on for the elISA application thus is mostly aimed at supporting the IsaWiki requirements, and to a lesser extent for general web page analysis.
We have run and are still running a series of tests to verify the correctness and solidity of the rules and the overall application we are working on. A first test was performed on almost 100 real web pages (in Italian and English), that have proved the overall trustworthiness of the elISA system. More pages are being checked weekly, as we plan to reach a number of 3/500 different web pages from all over the world.
Our tests have used a set of 19 rules aimed at identifying six different sections that are often found in real web pages, including content zones, layout zones (parts of the page without content, introduced as spacers between other zones), navigation zones (meant to contain navigational links within the site, rather than actual content), forms, logos and footers. Not all pages have identifiable parts of the page containing these elements, and many pages have many of each.
The experimental results are promising. As shown in fig. 9, the most important elements for the IsaWiki application are identified correctly.

	page part	present	identified	ignored	success
1	content	1262	1262	0	100.00 %
2	layout	5404	5305	99	98.17 %
3	navigation	2201	2198	3	99.86 %
4	form	147	147	0	100.00 %
5	footer	25	15	10	60.00 %
6	logo	141	44	97	31.20 %

fig. 9: experimental results with 97 web pages

The low results for the identification of footers and logos depend equally on our lack of consideration, so far, for those parts (we only included a few very basic rules), and on the number of different ways in which those zones are coded in real web pages.

Of course these are only preliminary results. The rules for footers and logos can be easily extended to improve the overall success rate, and the rules for content zones can be refined to exclude smaller text elements of no relevance to content. Furthermore, new categories for page parts can be devised and new rules can be built for their identification. In the next months we plan in fact to extend the set of rules used by the elISA application. More details about our experiments can be found at the URL http://tesi.fabio.web.cs.unibo.it/elisa.

6 Conclusions

In this paper we have proposed a new application field for structural analysis of web pages, i.e., the identification of content areas for in-place editing of web pages, as required, for instance, by the IsaWiki system.

For this reason we have created a rule-based analysis tool called elISA, that can identify content areas by applying rules expressed in a simple XML-based language. The whole architecture is very simple, revolving around an XSLT transformation whose rules are created on the fly by applying a metastylesheet on the elISA rules. One of the advantages of this approach is that the procedural part of the whole system is minimal (the cleansing of the web page, mostly) and that it can be ported with very little effort to a number of different operating environments.

The elISA rule language is simple and yet powerful, for it allows the expression of a large number of cases and patterns by means of the XPath pattern language. This also provides an incredible flexibility and power to the application with very limited implementation efforts.

The development of the elISA application is not finished yet. Besides adding a few more rules and polishing the syntax, we plan in the future to test our rules on a large number of real web pages, trying to fine tune the rules and add the odd new ones that would make manifest in the course of the test. Updated and complete information about elISA can be found at http://tesi.fabio.web.cs.unibo.it/elisa.

Acknowledgements

Special thanks go to Matteo Bagnasco that is performing tests on the rules of elISA. We would also like to thank all the masters and undergraduate students at the University of Bologna who took and are taking part to the IsaWiki project for their master thesis and final projects, respectively. They are: Nicola Bagnasco, Lauro Cottafavi, Mariano Desio, Gabriele Fantini, Stefano Monducci, Pietro Nanni, Simona Orselli, Stefano Rizzoli, Stefania Sangiorgi, Michele Schirinzi and Roberta Zeppilli. We also thank Microsoft Research for their partial support to this endeavor.

References

[1] "Amaya", W3C, http://www.w3.org/Amaya/.

[2] "Annozilla (Annotea on Mozilla)", http://annozilla.mozdev.org/

[3] Blood R., "Weblogs: a history and perspective", 2000, http:// www.rebeccablood.net/essays/weblog_history.html.

[4] Bieber M., Vitali F., Ashman H., Balasubramanian V., Oinas-Kukkonen H. "Fourth Generation Hypertext: Some Missing Links for the World Wide Web", *International Journal of Human-Computer Studies*, 47, 1997, pp. 31-65.

[5] Chen Y., Ma W., Zhang H., "Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices" *Proceedings of the Twelfth International Conference on World Wide Web, (WWW2003)*, May 20-24, 2003, Budapest, Hungary

[6] “ComMentor”, Standford University, http://www-diglib.stanford.edu/diglib/pub/reports/commentor.html

[7] Cunningham W., Leuf B., *The Wiki way*, New York, Addison-Wesley, 2001.

[8] Di Iorio A., Vitali F. Writing the Web, *Journal of Digital Information*, 5(1), 2004-05-27, art. n. 251.

[9] Gupta S., Kaiser G., Neistadt D., Grimm P., "DOM-based Content Extraction of HTML Documents" Proc. of WWW2003, May 20-24, 2003, Budapest, Hungary

[10] iMarkup, iMarkup Solutions, http://www.imarkup.com/products/imarkup_client.asp

[11] Kovacevic M., Diligenti M., Gori M., Milutinovic V., "Recognition of Common Areas in a Web Page Using Visual Information: a possible application in a page classification", *Proceedings IEEE International Conference on Data Mining*, 9-12 Dec 2002, 250-257

[12] Koivunen M.-R., "The Annotea Project", World Wide Web Consortium, 2001, http://www.w3.org/2001/Annotea/

[13] Mukherjee S., Yang G., Tan W, Ramakrishnan I.V., “Automatic Discovery of Semantic Structures in HTML Documents”, in *Proceedings of Seventh International Conference on Document Analysis and Recognition (ICDAR2003)*, Edinburgh, Scotland, August 3-6, 2003, p. 245-249

[14] Nelson T.H., *Literary Machines*, Sausalito (CA), USA, Mindful Press,

[15] Nanno T., Saito S. and Okumura M., “Structuring Web pages based on Repetition of Element", *Proceedings of the Second International Workshop on Web Document Analysis, (WDA2003)*, August 3, 2003, Edinburgh, UK, p. 7-10.

[16] Penn G., Hu J., Luo H., McDonald R., "Flexible Web Document Analysis for Delivery to Narrow-Bandwidth Devices", *6th International Conference on Document Analysis and Recognition, (ICDAR01)*, Sept 10-13, 2001, Seattle, The United States

[17] Vitali F., "Creating sophisticated web sites using well-known interfaces" in *HCI International 2003 Conference*, Crete (Greece), June 2003.

[18] Vitali F., "Functionalities are in systems, features in languages. What is the WWW?", In *IV Hypertext Functionalities Workshop, Seventh International World Wide Web Conference*, Brisbane, 14th April 1998, http://www.cs.nott.ac.uk/~hla/HTF/HTFIV/fabio.html

[19] Yee K., CritLink: Advanced Hyperlinks Enable Public Annotation on the Web, *Demo to the CSCW 2002 conference*, New Orleans, Dec 2002, http://zesty.ca/pubs/yee-crit-cscw2002-demo.pdf

[20] Yang Y., Zhang H., "HTML Page Analysis Based on Visual Cues", *6th International Conference on Document Analysis and Recognition, (ICDAR01)*, Sept 10-13, 2001, Seattle, The United States.

[21] Zhang H., Chen J., Shi J., Zhou B., Fengwu B., “Function-Based Object Model Towards Website Adaptation”, *Proceedings of the Tenth International Conference on World Wide Web, (WWW2001)*, Hong Kong, May 1-5, 2001, ACM Press, p. 587-596