Analysis and regeneration of hypermedia contents through Java and XML tools

David Mérida, Ramón Fabregat, Anna Urra, Antonio Bueno
Institut d'Informàtica i Aplicacions (IIiA)
Universitat de Girona (UdG)
Lluìs Santaló Av., 17071 Girona, SPAIN
+34 972 41 8475
david.merida@udg.es, ramon.fabregat@udg.es, anna.urra@udg.es, antonio.bueno@udg.es

Abstract

This paper presents a tool, for the analysis and regeneration of web contents, implemented through XML and Java. At the moment, the web content delivery from server to clients is carried out without taking into account clients' characteristics. Heterogeneous and diverse characteristics, such as user's preferences, different capacities of the client's devices, different types of access, state of the network and current load on the server, directly affect the behavior of web services. On the other hand, the growing use of multimedia objects in the design of web contents is made without taking into account this diversity and heterogeneity. It affects, even more, the appropriate contents delivery. Thus, the objective of the presented tool is the treatment of web pages taking into account the mentioned heterogeneity and adapting contents in order to improve the performance on the web.

Keywords: *Hypermedia System, Heterogeneous Networks, Content Adaptation, Adaptive Web System.*

1. Introduction

At present, the enormous heterogeneity in terms of types and capacities of access devices, bandwidth of the network and needs/preferences of the users are not taken into account by a server when providing web content that is rich in images, audio and video. The server will deliver the document requested even if the terminal used (WebTV, PDAs or mobile telephones) can not access these content due to the limitations of the display, of the storage capacities, of processing or of access to the network.

To solve this problem, different alternatives [1][2][3] have been developed that allow universal access to any type of material taking into account the heterogeneity of: user's preferences, client's devices capacities, access types, state of the network or/and current load on the server.

In [4] SHAAD[1] is defined as a system that takes into account the mentioned heterogeneity and tries to adapt dynamically or statically the available information and to deliver it in the most efficient way. It incorporates the concepts of adaptability, adaptively and dynamism, summarized from works carried out by Brusilovsky [5][6], and De Bra [7][8][9].

SHAAD integrates in an unique structure the necessary mechanisms for an appropriate contents delivery. The system is made up of 4 modules (figure 1):

- *Mechanisms for characterization of adaptation variables.*
- *Content Module*: to deliver the content requested.
- *Decision Engine*: the kernel of the system and the place in which the adaptation variables and the available content are evaluated, from which are inferred the mechanisms for delivering the material in the form that best suits the end user.
- *Adaptation Mechanisms*: to implement the adaptation mechanisms decided upon by the Decision Engine

The SHAAD modular structure allows an independent development of the parts that compose it.

The function of the *Content Module* is to deliver the requested content, either through a dynamic generation (from the unit elements that make up the web page, *on-line generation*), or by selection from the various different static versions of this content (previously generated, *off-line generation*). The selection of either option will depend essentially on the current load on the server.

The *Adaptation Mechanisms* is an independent module, and intimately related with the *Content Module*. This is, the *Adaptation Mechanisms* can be applied directly on the page surrendered by the content module to the decision engine or during the process *on-line*.

The present work carries out the description of a tool (*Content Analyzer*) dedicated to the analysis and

[1] SHAAD is Spanish acronym of "Adaptable, Adaptive and Dynamic Hypermedia System"

Proceedings of the International Conference on Information Technology: Computers and Communications (ITCC'03)
0-7695-1916-4/03 $17.00 © 2003 IEEE

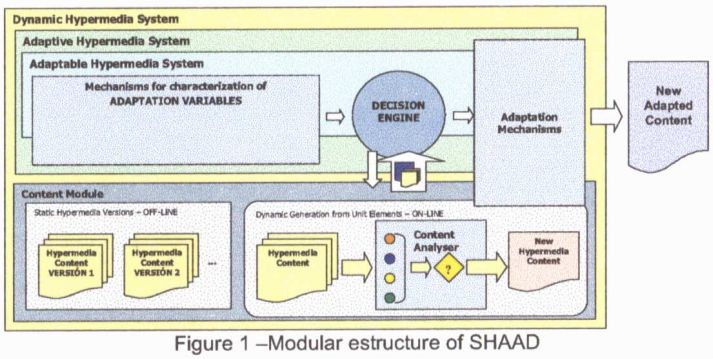

Figure 1 –Modular estructure of SHAAD

regeneration of the HTML content during this last process. This tool takes into account the mentioned heterogeneity.

This paper is organized as follows. Section 2 describes the implementation and processes of *Content Analyzer*. Section 3 describes how requested pages are readdressed to the *Content Analyzer*. Section 4 explains the user's interface that the system has. Finally, Section 5 describes conclusions and future works.

2. Content Analyzer

The *Content Analyzer's* objective is to carry out an analysis and regeneration of requested web pages, adapting its content to a series of input variables that are defined for:

- User's characteristics and preferences
- The characteristics of the client's device
- The current state of the net
- The access type to the net

This Content Analyzer is part of SHAAD content module, and it is triggered when the load on the server is low.

XML [10] and XSL were used for the implementation[2] of *Content Analyzer*. XSL uses XML notation and is a language for expressing stylesheets. An XSL stylesheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document that uses the formatting vocabulary.

XSL consists of three parts: *XSL Transformations (XSLT)*[11], a language for transforming XML documents, *XML Path Language (XPath)*, an expression language used by XSLT to access or refer to parts of an XML document, and *XSL Formatting Objects (XSL-FO),* an XML vocabulary for specifying formatting semantics.

The processes to take into account in the *Content Analyzer* are (figure 2):

- HTML to XHTML transformation
- Adaptation profile obtaining
- Adapted HTML obtaining

2.1- HTML to XHTML transformation

Actually exist different browser technologies, some browsers run internet on computers, and some browsers run internet on mobile phones and hand helds. Each one handle in a different way the "bad" marked up HTML documents. For example, the following HTML could present some problem for them:

```
<html>
<head>
<title> Bad HTML code </title>
<body>
<h1>Bad HTML
</body>
```

XHTML [12] consists of all elements in HTML 4.01 combined with the syntax of XML. HTML was designed to display data and XML was designed to describe data. XML is a markup language where everything has to be marked up correctly, which results in "well-formed" documents

HTML to XHTML transformation solves problems that arise from mal-formed HTML files. Some of the most important characteristics of XHTML are:

- *XHTML elements must be properly nested.* In HTML some elements can be improperly nested within each other; in XHTML all elements must be properly nested within each other.
- *XHTML documents must be well-formed.* All XHTML elements must be nested within the <html> root element. All other elements can have sub (children) elements. Sub elements

[2] The implementation through Java was also studied. This language has many classes defined that allows to generate code easily; but this possibility was discarded, due to the additional maintenance difficulty induced by Java development.

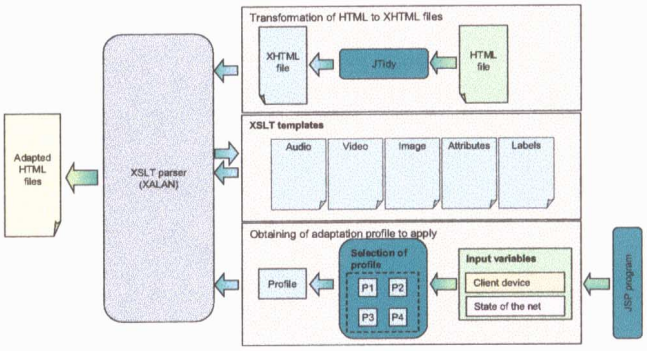

Figure 2–Analyzer of Contents

must be in pairs and correctly nested within their parent element.

- *Tag names must be in lowercase.* This is because XHTML documents are XML applications and XML is case-sensitive. Tags like
 and
 are interpreted as different tags.
- *All XHTML elements must be closed.* Non-empty elements must have an end tag, for example, the following tag *
This is a list item*, must be closed like this *
This is a list item *. Empty elements must also be closed. They must either have an end tag or the start tag must end with />, for example, the tag *<img src="happy.gif" alt="Happy face">*, should be written **.
- *Attribute values must be quoted.* For example, the tag *<table width=100>* should be written *<table width="100">*.

Previously and just for one time, HTML to XHTML transformation is carried out through JTidy [13]. It is a open source code program that allows us to analyze HTML files and to obtain a well formed XHTML. Then, we can apply XSL and its tools to make the transformation.

2.2- Adaptation profile obtaining

According to the value of the input variables, an adaptation profile is assigned. This profile indicates the adaptation type to apply to requested web page.

In order to check the operation of the content analyzer, the user specifies the values of these variables by means of web interface. The input variables considered are characteristics of the client's device and state of the network. Table 1 shows possible values of each one of these variables. Table 2 shows profiles defined by combination of considered values for the screen area and the atate of the network. Finally, Table 3 shows possible adaptations considering profiles and multimedia elements available.

In the current implementation, different static versions of each multimedia object exist in the server to simulate the adaptation mechanisms. When a profile is applied, the directory that contains the elements to send is determined.

2.3- Adapted HTML Obtaining

The adaptation of requested HTML files is carried out through the XSLT templates that are applied on obtained XHTML files (section 2.1)

From these XHTML files and using XSL / XSLT, transformations can be implemented in two ways:

- To define, on an unique XSLT template, all profiles that can be applied to the document XHTML. This template will include so many conditional branches as profiles exist. To consider a new profile, it means to introduce a new branch in this template.
- To define individual templates, one for each media object and considering profiles inside these templates. To take into account a new profile, it means to modify each template individually. To consider a new media object, it implies to generate the corresponding XSLT template. This treatment, carried out on individual templates, allows an easier maintenance. It has been selected for our proposal.

Proceedings of the International Conference on Information Technology: Computers and Communications (ITCC'03)
0-7695-1916-4/03 $17.00 © 2003 IEEE

Considered variable	Value possible
Screen area	High
	Medium (640x480)
	Low (PDA)
Color	Color
	Gray Scale
Video	Yes
	No
Sound	Yes
	No
State of the network	Free
	Semi-busy
	Busy

Table 1–Variables that determine the profile type to apply

Screen Area	State of the network		
	Free	Semi-busy	Busy
High	Profile1	Profile2	Profile4
Medium	Profile2	Profile2	Profile4
Low	Profile3	Profile3	Profile4

Table 2– Activation values of profiles

Media Object	Profile1	Profile2	Profile3	Profile4
Image	Original with the corresponding color treatment (*)	Reduced image in half and with the corresponding color treatment	Reduced image to the fourth part of the original and with the corresponding color treatment	Multimedia objects are not placed in the final document
Video	If it has the necessary software the original it is sent with the corresponding color treatment	Sequence of 10 images on which the image treatment of Perfil2 is carried out	Sequence of 5 images on which the image treatment of Perfil3 is carried out	
Sound	If it has the necessary software the original is sent	If it has the necessary software the original it is sent	If it has the necessary software the original it is sent	
(*) Color treatment means that the image can be in Color or Gray Scale				

Table 3 –Multimedia objects that are sent according to the defined profile

Now, to apply an XSLT template on XHTML file and to introduce parameters, a stylesheet processor should be used, for example SAXON[14] or XALAN[15]. This processor is used to check that the document is well formed and validates against the corresponding DTD and it is in charge of applying the stylesheet XSL, etc.

SAXON, is a processor written in Java with its own parser, although it can use an external one. XALAN, from Apache XML project, is a free tool written in Java that needs a Java virtual machine (JVM) and an XML parser to work. Xerces is the used parser and the corresponding file (xerces.jar) it is included in the distribution. To implement our analyzer we have used XALAN since we use an Apache server[3].

3. Readdressing of requested pages

The analyzer's execution is carried out on-line. When receiving the petition, the requested page is analyzed. If the petition is a HTML file, the analyzer is executed and the page is obtained adapted to the certain profile.

In order to execute the analyzer, the module *mod_rewrite* of Apache server is used [16]. It provides virtually all of the functions one would ever need to manipulate URLs, and its functionality is highly generalized. Consequently, *mod_rewrite* can be used to solve all sorts of URL-based problems. It has been considered two directive of this module:

- RewriteEngine. This directive enables or disables the runtime rewriting engine. If it is set to off, then mod_rewrite does no runtime processing
- RewriteRule. This directive allows to write the rules that will readdress the requested URLs. The basic sintax of this directive is:

  RewriteRule url-pattern url-new

  The *url-pattern* is a regular expression that is applied to the current URL. The current URL may not be the original requested URL, because any number of rules could have already matched and altered it. The *url-new* argument is the string that is substituted for the original URL matched by the url-pattern.

The module activation *mod_rewrite* of the Apache is carried out through the file httpd.conf. The next code shown part of this file, in which the *mod_rewrite* module is activated, it include the rules to apply with the *RewriteRule* directive:

```
<IfModule mod_rewrite.c>
  RewriteEngine On
  RewriteRule /adaptador/web/([^\.]+)/([^\.]+)\.(html|htm)
          /adaptador/jsp/adaptar.jsp?&PATH=$1&XML=$2 [R]
</IfModule>
```

Where:

[^\.] : any text that doesn't contain point + /

[3] Apache 2.0 on a Pentium II running Linux REDHAT 7.2, with 192 MB main memory, and 100-BaseT network connection has been used

PATH : requested URL address, without the name of the HTML / HTM file

XML : name of requested page –HTML / HTM

If the requested page coincides with some of rules specified through the *RewriteRule*, a JSP program is executed that will shoot the processor of style sheets to carry out the adaptation (Xalan). The following command shows the sentence of Xalan execution.

java org.apache.xalan.xslt.Process -IN prova.xml -XSL prova.xsl –OUT prova.out -PARAM perfil 2

4. User's Interface

The *Content Analyzer* has two interfaces. Figure 3 shows the interface that allows assigning the input variable values considered. The assignment is carried out through given values list. With values confirmation, any requested page will be adapted according to these values.

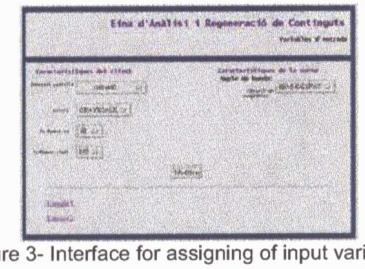

Figure 3- Interface for assigning of input variables

Figure 4 shows the error screen that is visualized when some execution error takes place. This screen allows to examine the *Content Analyzer*'s behavior and to identify through the error message the problem to solve. It is useful for the maintenance of the tool.

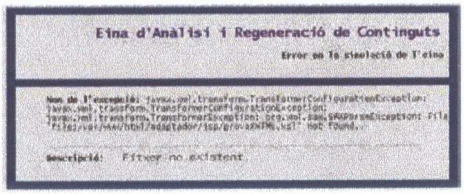

Figure 4- Error screen

5. Conclusion

The main advantage of the proposed approach is the simplicity. The content analyzer shows the feasibility of adapting the requested web page according to a series of input variables.

XML and Java are a simple way to implement the analyzer, and therefore a quite simple way to maintain it.

Future works has been pending to include more elaborated profiles, and to obtain an integral support for adapted content delivery through this tool.

Acknowledgment

This work has been partially financed by CICYT TEL-99-0976 and Galecia project (UE-SOCRATES-MINERVA).

References

[1] N. Shaha, A. Dessai, and M. Parashar," Multimedia Content Adaptation for QoS Management over Heterogeneous Networks," *Proceedings of the International Conference on Internet Computing (IC 2001)*, Nevada, USA, pp 642 - 648, Computer Science Research, Education, and Applications (CSREA) Press, June 2001.

[2] S. Campadello, H. Helin, O. Koskimies, P. Misikangas, M. Mäkelä, K. Raatikainen, "Using Mobile and Intelligent Agents to Support Nomadic Users", 6th International Conference on Intelligence in Networks (ICIN2000), Bordeaux, France, 2000.

[3] Y. Yang, J. Chen, H. Zhang, "Adaptive Delivery of HTML Contents", *9th International World Wide Web Conference – The Web: The Next Generation*, Amsterdam, 2000.

[4] D. Mérida, R. Fabregat, J.L. Marzo, "SHAAD: Adaptable, Adaptive and Dynamic Hypermedia System for content delivery", *Workshop on Adaptive Systems for Web Based Education (WASWE2002)*, Málaga (España), 28 may 2002.

[5] P. Brusilovsky, "Methods and Techniques of Adaptive Hypermedia", *User Modeling and User-Adapted Interaction, Vol. 6, pp. 87–129, Kluwer academic publishers*, 1996.

[6] P. Brusilovsky, "Adaptive Hipermedia", *User Modelling and User-Adapted Interaction 11: 87-110, Kluwer Academic Publishers*, Netherlands, 2001.

[7] P. De Bra and L. Calvi, "Towards a Generic Adaptive Hypermedia System", *Proc. Second Workshop on Adaptive Hypertext and Hypermedia*, Pittsburgh, pp. 5–11, 1998.

[8] P. De Bra, "Design Issues in Adaptive Web-Site Development", *Proceedings of the 2nd Workshop on Adaptive Systems and User Modelling on the WWW*, Canada, 1999.

[9] P. De Bra, G.J. Houben, H. Wu., "AHAM: A Dexter-based Reference Model for Adaptive Hypermedia", *Proceedings of ACM Hypertext'99*, Darmstadt, pp. 147-156, 1999.

[10] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", *W3C Recommendation*, 6 October 2000. http://www.w3.org/TR/2000/REC-xml-20001006

[11] J. Clark, "XSL Transformations (XSLT) Version 1.0", *W3C Recommendation*, 1999. http://www.w3.org/TR/xslt

[12] S. Pemberton et al., "XHTML 1.0: The Extensible HyperText Markup Language", *W3C Recommendation*, Jan 2000. http://www.w3.org/TR/2000/REC-xhtml1-20000126

[13] http://www.w3.org/People/Raggett/tidy

[14] Saxon Version 6.3. http://users.iclway.co.uk/mhkay/saxon/saxon6.3/index.html#Scope

[15] Xalan-Java Version 2.3.1. http://xml.apache.org/xalan-j/index.html

[16] Apache HTTP Server Version 1.3, *mod_rewrite* Module Information. http://httpd.apache.org/docs/mod/mod_rewrite.html