A MODULAR ADAPTATION FRAMEWORK FOR SINGLE SOURCE PUBLISHING

Thomas Springer, Steffen Göbel
Dresden University of Technology,
Department of Computer Science,
Institute for System Architecture,
Mommsenstr. 13,
D-01062 Dresden, Germany,
{springet, goebel}@rn.inf.tu-dresden.de

ABSTRACT

Multi-device provisioning of web-based services is a challenging problem for service providers because of the heterogeneity of their execution environment. To support this heterogeneity, services have to adapt to the varying capabilities of used devices, markup-languages, connectivity, user preferences and access context. In this paper we describe general adaptation mechanisms and propose a proxy-based adaptation framework which combines the advantages of manual and automated adaptation. The framework is based on a generic document description language (DDL) allowing the manual addition of semantic meta information at design time. This meta information is used to control the automated adaptation process at runtime. The framework supports the addition, removal and substitution of modular designed adaptation mechanisms.

KEYWORDS

adaptation, fragmentation, transcoding, XML, proxy, mobile computing

1. INTRODUCTION

The language of today's web is HTML. It has been developed for desktop computers and fast and stable wired networks. Extended features as Frames, Cascading Style Sheets (CSS), and active content such as JavaScript or Java as well as proprietary extensions are supported by state-of-the-art desktop browsers only. Most of the browsers on mobile devices available today only support a subset of the current HTML4 standard [13], are limited to older versions of HTML or even use a device specific markup-language (e.g. WML [15] or cHTML [18]). This implies that documents depending on browser or device specific features cannot be presented correctly or are not displayable at all if devices do not support these features.

Further problems occur because of the heterogeneity of hardware and software features of mobile devices (e. g. input and output capabilities, memory, processing power, operating system, and supported multimedia formats). The capabilities of wireless and wired communication systems differ as well (e. g. bandwidth, delay, error rate and costs). Beyond the technical parameters, personalization is an important factor for the usability of services. User preferences vary as well as the context of the service access (e. g. time, location, activity and role of the user). Therefore, the content and behavior of web-based services have to be adapted to perform well within this highly heterogeneous execution environment.

In general, web-based services can be adapted in two ways: manual and automated. *Manual adaptation* leads to multiple dedicated versions of web documents and provide the best results in usability and design. The disadvantage is the (in most cases) unreasonable effort to create and maintain the consistency of web content. A new version of the content has to be created for each new device or class of devices. *Automated adaptation* in contrast requires little effort to create and maintain web content, but in most cases usability and design of the resulting documents are unpleasant.

In our approach we have chosen a combined solution. We propose a generic Document Description Language (DDL) based on XML which enables the separation of structure, representation, and content of web

documents. It adopts several concepts from UIML [1], XForms [5], and XML Schema [3]. The DDL allows the manual addition of semantic meta information at design time. For example, meta information may indicate alternative representation of semantically one element. Furthermore, elements can be declared to be optional and may be omitted on devices with insufficient resources. This meta information is used to control the automated adaptation process at runtime. The adaptation process comprises device identification and classification, session management, data input validation, dialog fragmentation, transcoding, and mechanisms to adapt the content to the capabilities of the user device and connectivity. Therefore, we provide a proxy-based framework which enables the addition, removal and substitution of modular designed adaptation mechanisms according to information of the execution environment.

The reminder of this paper is organized as follows: In section 2 we describe general adaptation mechanisms for web documents. Section 3 introduces our document description language DDL. Implementation details of our adaptation framework and an evaluation are covered in sections 4 and 5. We conclude the paper with a discussion of related work (section 6) and an outlook to the future (section 7).

2. ADAPTATION MECHANISMS

The term adaptation describes the ability of a system to react on changes within its execution environment. In the context of web-based services, the structure, content data, and the transmission of the data are subjects for adaptation. In this section adaptation mechanisms for web content will be discussed in general.

2.1 Adapting the structure of web documents

Web documents consist of internal (e. g. text and GUI elements) and external elements which are linked to the document (e. g. images, audio, and other media objects). If the links do not contain any information about the referenced elements (as it is the case for HTML), these elements can be only involved into the structure adaptation by using heuristics. If information about linked elements is available, a finer granularity of adaptation is possible. For instance, lossy operations could be performed according to a priority value describing the importance of the elements for the look and feel or the semantic within the document. For instance, in a document adapted to a small display size, lowly prioritized elements should be elided while highly prioritized elements are kept unchanged in contrast to reduce the size of all images equally.

A loss-less approach for structure adaptation is fragmentation. It prevents the elision of elements by distributing them among several pages which can be navigated via links. The relationship between elements (e. g. an image and its caption, or a text field and its description) can be expressed by the definition of atoms (which are indivisible) and groups of atoms (which are semantically related but can be divided if necessary). As proposed in [2], tables can be transformed into one sub-page per table cell, in top-down, left-right order.

2.2 Adapting the elements of web documents

The single elements of web documents can be adapted by converting their properties (e. g. resolution and color depth of an image) and data representations (i. e. file format). Furthermore, the quality is adjustable by applying lossy compression. The replacement of elements is a very powerful mechanism to reduce the amount of data or to overcome incompatibilities. A wide range of mechanisms is applicable which all change the type of the element (e. g. speech to text or video to image sequence) while keeping as much of the semantics of the original element as possible. Decisions for the adaptation of the described mechanisms should take the following questions into account:

1. What properties of the element are adaptable?
2. What are the results of the adaptation?
3. What additional information is necessary for the adaptation process?

Structured text: Text is normally structured into several parts (e. g. title, headings, sections, abstract and meta information such as author, creation date, or keywords) describing the semantics of text within a document. To adapt a text, only certain elements can be used to create new views. For instance, a table of contents

can be created out of the headings, the abstract could be extracted together with meta information about the author, or a certain part of the document could be selected according to given keywords. Furthermore, the first x words or the first sentence of a section can be presented to create an overview of the section (as proposed in [2]). The goal of adapting structured text is to provide several views to get a quick overview of the whole document and to fragment large documents according to a given display size. The reduction of data volume is only important for devices with very restricted main memory such as mobile phones. A fragmentation of large documents into pages, which fit to the display size, would provide a sequential viewing of the pages of the document. This mechanism reduces the amount of data transferred over the network if not all pages of the document are viewed (known as lazy evaluation [16]).

To adapt text documents as described, meta information about the structure is required. This information can be explicitly added or extracted from unstructured text by heuristics (see [14]). Additional keywords given by the user can be used to search and extract interesting sections.

Images: Images and text are the most frequently used elements in web documents, but especially the image size has a big influence on the rendered document size. The adaptable properties of an image are the resolution, color depth, a quality factor expressing the information depth of the image (e. g. the compression factor for JPEG images), and the file format. The main goal of image adaptation is to reduce the file size while keeping as much of the information as possible. According to the reduction of the amount of transferred data, thumbnails with a link to the original image can be created leaving the decision of the transfer to the user. *Sections* of an image can also be extracted to give the user a preview of an image[6], [8]. If the available bandwidth is too low or the display size is too small, images can be replaced by a textual description.

2.3 Adapting the transfer of web documents

Typically the last link to mobile devices is a low bandwidth, high latency, error-prone, and high cost network connection. A proxy approach (see for instance [6], [8] and [21]) enables the adaptation of documents before the transfer. Functions usually performed by the client device can be done by the proxy. In the case of network errors, the proxy can receive and cache messages and data for the client to prevent data loss. The proxy also allows the installation of adaptation mechanisms as done in the approach described in this paper.

3. DOCUMENT DESCRIPTION LANGUAGE

For the description of documents we had to decide to define a new markup language or to use an existing one. The analysis of existing approaches showed that most of the available markup languages (e.g. HTML or WML) are device specific. To support arbitrary devices, a chosen language has to be transformed into a suitable, possibly different markup language. Due to the lack of meta information about the semantic structure of the document, however, the mapping to other markup languages is based on heuristics only, is often ambiguous and for some of the elements impossible. But a language with only simple elements, which can unambiguously be mapped to all devices, cannot exploit the capabilities of more powerful devices.

Describing the documents with a generic language, structure information as well as additional meta information for the adaptation can be included. This results in better designed and usable documents after the adaptation. The approach supports multiple target languages and is extensible to further languages by introducing new transformations. As a disadvantage, existing documents have to be translated into the generic language before they are accessible.

Because of the identified advantages we decided to use a generic markup-language. To explore the possibilities of automated adaptation controlled by manually attached meta information, a language is required which is simple and compact, but also functional, powerful, and extensible. Because from the existing languages no one met all of our requirements, we decided to define a new language. The *Document Description Language* (DDL), hereby introduced, fulfils these requirements. It follows the concept of decoupling of structure, representation and content, and the concept of inheritance. Furthermore, constraints on user input in forms can be specified which allow automated input validation by the adaptation software.

DDL is an XML-based meta language. It describes a structure of abstract elements. Properties can be assigned to each element. Furthermore, a “test”-attribute can be defined for some of the elements which enables

the specification of conditions for their inclusion. A condition is defined by an XPath-expression applied to the client profile. The structure of a DDL document is shown in figure 1. The root element is **<ddl>**. It may be followed by optional include statements, header information, and datatype definitions. The document section describes the visible document itself. Parts, classes, and content can be defined inside or outside the document section, but only the part elements inside will form the web page. All other elements are library elements that can be used to for inheritance.

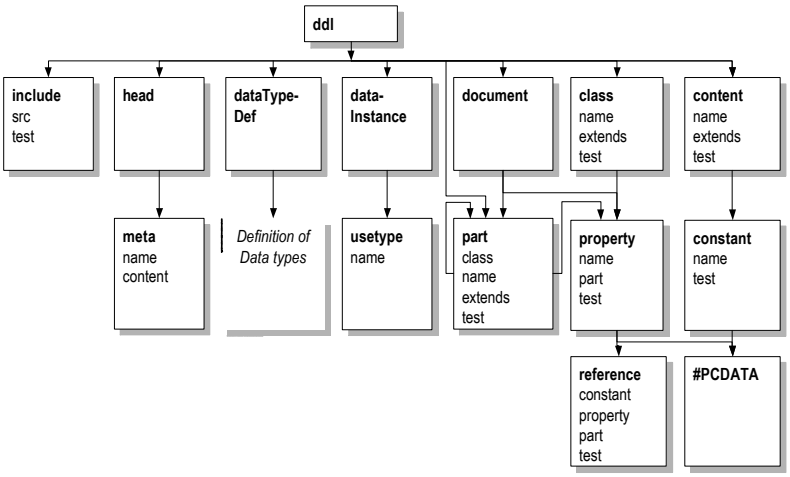

Figure 1: DDL syntax

With the **<include>** statement external DDL files can be included which easily allows the creation and reuse of DDL libraries. Within the **<head>** section meta information of the document (e. g. author or creation data) can be described. There is at most one <head>-Element per DDL document. The data definition section allows the definition of data types and data instances which are used to validate input of web forms. From a set of basic data types complex data types can be derived within **<dataTypDef>** elements. Several restrictions (e. g. min and max value for integer data) can be assigned to each built-in basic data type allowing precise specification of valid inputs. Type information can also be used to optimize the presentation of input elements. For instance, a calendar could be displayed to prompt for a date. Each data type is assigned a unique name which can be used to define the type of **<dataInstance>** elements. Input elements of web forms can be bound to data instances via the unique names of data instances. By this means, the user interface and the data is separated which enables the adaptation of the representation of input elements.

Within the **<document>** section the structure of the document is described. There is at most one <document>-Element per DDL document. As mentioned it comprises all elements forming the actual web page. It is formed by parts, classes, and content elements.

<part>-elements are used to model the abstract structure of a document. They can be nested and may have properties assigned to them. The optional "extends"-attribute refers to another part by a unique logical name to inherit properties from. By means of an optional "class"-attribute a class can be assigned to a part. If a part belongs to a class and additionally inherits from another part, properties of the class have higher priority than those inherited from another part and therefore override them.

The **<class>** element defines a class of parts. It comprises a set of properties. All properties of a class are adopted by its instance parts. Analogously to parts, the optional "extends"-attribute can be used to define class inheritance. By means of **<content>**-elements DDL realizes decoupling of structure and contents. The <content>-element comprises a set of data items (cf. <constant>-element) that can be referenced by <property>-elements or other <content>-elements.

<property> elements are used to assign properties (styles for the presentation or abstract properties) to parts or classes. The semantics of the properties is defined separately. By this means, future extensions may be developed without the need to change the syntax (DTD) of DDL. Only the DDL renderers (usually in the form of XSLT style sheets), for the adaptation to device specific languages, must be extended or adapted to be able to interpret new properties.

Parts are assigned semantic types specifying the interpretation of the particular <part>-element by the renderer. Currently we have defined a set of parts to create web documents. Within a **container** part arbitrary

parts can be grouped together to specify a certain layout or to define atoms for the fragmentation. Structured text can be described, for instance, by the paragraph and abstract part. Further parts exist to define attributes of text, images, and tables. To create forms, we have defined a form part, part elements for user input related to HTML forms (e. g. textinput, radiogroup or checkbox), and a submit part to finish a form.

4. ADAPTATION FRAMEWORK

With the development of our adaptation framework, we intended to apply most of the mechanisms described in section 2 to adapt web content. We used the open source HTTP proxy Muffin [12] as basis for the implementation of the prototype. Muffin is Java based and enables the modification of HTTP requests and responses by means of filter chains. Furthermore, the adaptation framework makes use of the Xalan-XSLT-processor [19] as well as the Xerces-XML-parser [20] by the Apache-Group. The adaptation is performed via a chain of filters (fig. 2) whereby a filter is simply a Java class implementing one or more of the Muffin Interfaces HTTPFilter, RequestFilter, and ReplyFilter.

The adaptation of web content uses information about the client device, the network connection, and the user preferences. The proxy determines this information at the beginning of processing a HTTP request in the **ClientRecognizerFilter**. A user has to logon to the proxy with username and password in order to assign a user profile to the connection (**UserRegistryFilter**). The client and network profiles are determined by means of the User-Agent HTTP header field or a transmitted CC/PP profile [11]. The latter way provides much more and more accurate information about the client device and network connection. Unfortunately, CC/PP is not supported by web browsers yet. Therefore, we emulated a CC/PP capable web browser by means of a client-side HTTP proxy inserting a CC/PP profile into the HTTP header.

The sequence of filters in the request processing chain is determined by a configuration file. Muffin processes the configuration information and controls the successive execution of the filters. Each filter has a `test` method that determines by a boolean return value if this filter should be applied in the current adaptation process. For instance, only requests from WML clients require the invocation of the WMLCompilerFilter (cf. fig. 2).

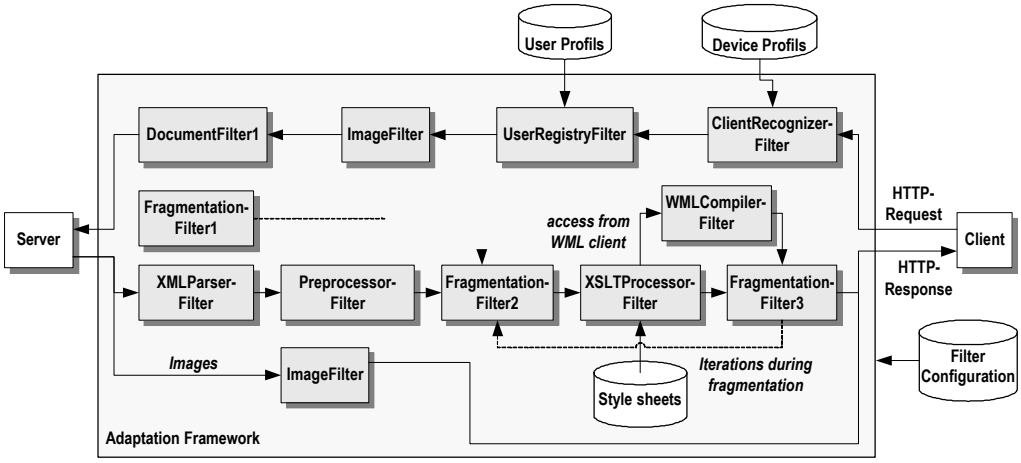

Figure 2: Filter chain of the adaptation framework

The adaptation process may include iterations, i. e. a loop in the filter chain. To allow for loops, a filter may optionally determine its successor. An example of a loop is the iterative fragmentation of documents by FragmentationFilter2 and FragmentationFilter3 (cf. fig. 2).

Currently we have implemented a number of filters empowering the adaptation framework to support the transformation of a DDL document into a device specific representation. Additional functionality can be easily added by implementing extra filters. In the following we give a brief overview of some filter classes and their functionality. All filter classes inherit some common functionality from the abstract superclass FilterSupport.

DocumentFilters: These two filters are responsible for the structural analysis of a document and the generation of a new presentation. DocumentFilter1 retrieves a requested paragraph generated by DocumentFilter2 from the local cache. This happens when a user chooses a particular section from the table of content or a curtailed presentation of a paragraph. DocumentFilter2 processes a document according to the definitions in the user profile. This includes displaying or hiding of particular meta information or abstract of a document, creation of a table of contents from section captions with links to section text, or the reduction of sections to the first sentence. Removed data is stored in a local cache to enable later retrieval by future client requests in conjunction with DocumentFilter1.

ImageFilter: This filter adapts images within a document according to user preferences and properties of the client device. First it checks if a particular image needs to be adapted or can be transmitted to the client unchanged. Then the ImageFilter replaces any image within a document by a adapted image and a link to the original image. Additionally, it converts images, e. g. BMP into JPEG or WBMP, high resolution into low resolution, or full color into grayscale. Several format specific parameters can be specified, e. g. the "quality"-parameter for JPEG images or the "interlaced"-parameter for PNG images.

PreprocessorFilter: This filter preprocesses a DDL document to resolve all external references and inheritance hierarchies. This results in a simplified DDL document with single <document>-block. By this means, the style sheet-based transformation is eased. Even the preprocessing may be style sheet-based. However, as this process is very time consuming, we decided to use a DOM-based transformation in Java.

XSLTProcessorFilter: This filter transforms a preprocessed DDL document into a device specific format (HTML, WML). The transformation is based on XSLT style sheets. The style sheets have access to the information in the HTTP request and the context of the processing environment. The information is mada available to the style sheets as XSLT parameters.

Fragmentation filters: These three filters work together to perform the document fragmentation in case the client's restrictions forbid the particular document to be displayed as a whole. Besides the actual fragmentation they are responsible for the user input validation and they store input data until the final dialog part is completed.

The FragmentationFilter1 manages the caching of document fragments and the fragment by fragment delivery to the clients. Furthermore, it stores input data until forms within the document are completed. FragmentationFilter2 does the actual fragmentation of a document if it has exceeded the resource restrictions of a device. First, the document is fractionalized into the smallest indivisible parts. Then these parts are combined to as few as possible fragments that still meet the resource constraints of the client. A more detailed description of the fragmentation algorithm can be found in [9]. Finally, FragmentationFilter3 checks if the size of the rendered document exceeds the resource restrictions of a particular device. If this is true, the filter invokes FragmentationFilter2 again to trigger another fragmentation iteration.

WMLCompilerFilter: WAP devices do not process a textual WML document but a compact binary representation (binary WML). Therefore a WAP gateway, an intermediary between the server and the WAP device, compiles the textual into the binary representation. However, this means the memory restrictions of the device do not apply to the size of the textual WML document but to the size of the compacted version. To check if a WML document fits the resource restrictions of the client, a WMLCompilerFilter is interposed between the XSLTProcessorFilter and the FragmentationFilter3 to perform the conversion to binary WML.

5. EVALUATION

To validate the functionality and applicability of DDL and our adaptation framework, we have developed a railway information system and an online newspaper as sample applications. The first example is a representative of dynamic and interactive web applications whereas the latter contains more structured and static content. By the means of these two examples, we want to illustrate some of the adaptation capabilities of our transcoding framework. Due to the limited space of this paper, there is only a screenshot of the railway information system.

The sample railway information system enables connection querying and ticket ordering in a fictitious railway company (cf. fig. 3). The comparison of the representations for desktop and WAP browsers shows many differences: The WAP representation omits the menu, and the two advertisement sections are replaced by a short text. The table with the timetable for the connections is transformed into a list with pairs of table

column name and table cell value on a WAP browser. Finally, much layout information gets lost due to the limitations of WAP.

The sample online newspaper tries to imitate an existing online newspaper but adds adaptation capabilities and is described by DDL. On desktop computers it has a classical three columns layout: a narrow left column with a topic list, a broad middle column containing either selected important articles with images and short text or a single article with full text, and a narrow right column with weather forecast, stock exchange information, and user surveys. On top of the page there is title banner.

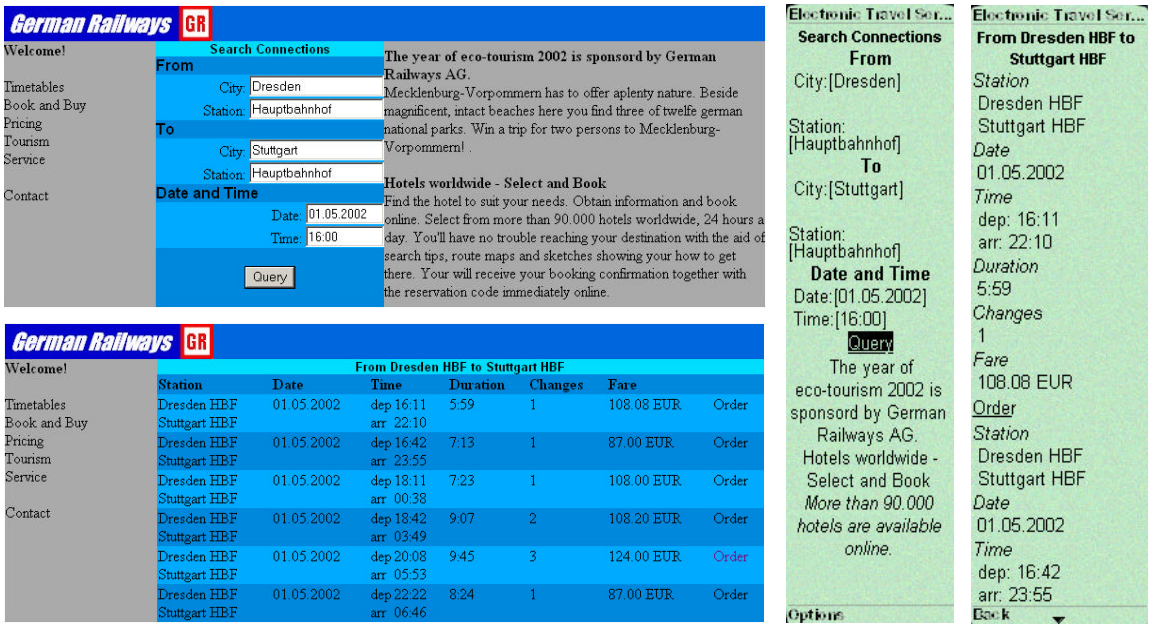

Figure 3: Railway Information System: desktop browser (left), WAP browser (right)

When using a PDA or mobile phone to display the same web page, several adaptations are automatically performed by the adaptation framework: The complex page layout is split up into a topic menu, a list of articles, and a complete article. Each of these parts is displayed on a separate page on the device. Image sizes are reduced to fit to the device display. If necessary the image format is converted, for example when using WAP mobile phones. Only the first sentence of each section is displayed in a long text article. The user can access the remaining parts by means of a link displayed at the end of each curtailed section. Unstructured long texts are also split up into several parts if the whole text would exceed the memory capacity of a device.

The user can specify in his user profile that he do not want to receive images at all. In this case, the adaptation framework replaces all images by an alternative text (which has to be added to the image within the DDL document) and adds an link to the image. This is especially useful if the user has to pay the amount of data transferred.

In order to evaluate our adaptation framework we have measured the performance for processing a complex DDL document. Thereby the major share of the processing time (ca. 80%) is consumed by the XSLTProcessorFilter and the DDLPreprocessor-Filter.

6. RELATED WORK

Because of the popularity of web access as well as its importance for business applications, many research projects have worked on the problem of adapting web content to heterogeneous mobile devices and wireless network connections. Most of the projects use HTML as source format. Using HTML all existing web documents can be accessed without changes. But in practice most HTML documents mix content and presentation. In contrast to HTML, in DDL and other XML-based approaches structure is explicitly described. Furthermore, because of using heuristics and definitions to control the adaptation process, its results are often

unpleasant and sometimes unusable. For instance Digestor [2] uses heuristics in addition to tag information to extract structure information. Our approach uses the DDL as source format to manually add semantic meta information at design time. This allows the designer to control the mechanisms used for adaptation which clearly distinguishes our approach from the HTML-based ones.

In Wit [16] web documents are described by linked hyperobjects. The structure of these hyperobjects reflect the hierarchical structure of documents (e. g. structured text is represented by separate objects linked together). The links are annotated by meta information such as the relative priority or the probability of access. This meta information is used to control caching, prefetching, and data reduction of the objects to improve the effects of adaptation. Wit focuses in contrast to our approach on the adaptation of the communication over wireless networks and supports Palm PDAs only. Especially it does not take device properties as display size into account.

M-Links [14] and [4] aim to change the structure of web documents to support small display sizes. In M-Links browsing is divided into navigation and using a service attached to a link. Therefore, the document structure is changed to a list of links which are partly extracted from the HTML document by heuristics (e. g. a phone number which can be dialed by following the link). In [4] structured text within a document is partitioned into semantic textual units which can be presented with different levels of content to the user. Both approaches use HTML as source format and aim to improve the navigation with small displays. [4] focuses especially on pen based input. Our approach supports small display and memory sizes by the fragmentation of documents. The fragmentation is controlled by device profiles and meta information which makes it more general while producing comparable results to the concepts of [14] and [4].

XDNL [10] is an XML-based approach to explicitly describe the document structure as a tree of leaf documents and navigation paths through the tree. The leaf documents represent displayable portions of information. Especially at different levels of the tree, the leaf documents contain different amounts of information. XDNL focuses on the improvement of navigation through large documents. This is a certain aspect of the DDL and we investigate what ideas of XDNL are adoptable.

The Top Gun Wingman Browser [7] introduces a modular concept of adaptation within a proxy which contains adaptation modules with clearly separated tasks. Filters for transforming and presenting images, HTML, and ZIP archives are implemented. The Browser supports only a special class of PDAs and processes HTML documents while our approach provides a generic solution for arbitrary devices. The Pythia system [6], [8] also uses a proxy architecture for the on-the-fly adaptation of services and data with focus on data specific lossy compression and refinement which is supposed to be much more efficient than loss-less compression. Similar lossy mechanisms are applied in our approach but integrated with additional mechanisms which work on the structure of the document.

IBM WebSphere Transcoding Publisher [17] uses a filter architecture similar to our framework. However, the Transcoding Publisher does not support loops within the filter chain. The main difference is the use of HTML in WebSphere in contrast to our DDL approach. Together with the flexible extendable architecture of our framework, which provides structure and content adaptation, the combined approach of manual and automated adaptation is the main contribution of our work. Furthermore, we take context information (e. g. device and user profiles) into account which enables the personalized adaptation of web services.

7. CONCLUSION AND FUTURE WORK

In this paper we have presented a modular adaptation framework supporting heterogeneous devices. It integrates several mechanisms for the adaptation of web documents to the special properties of mobile devices. We have introduced the device independent markup language DDL for the description of documents and forms. It contains additional meta information to improve the results of the automatic adaptation process. The integration of author knowledge into DDL also increases the usability of the generated device specific markup languages. Existing contents available in HTML cannot be used directly as input sources for the adaptation as a trade-off, but we plan to develop tool support for the transformation of HTML into DDL.

The evaluation of the framework has proven that the XSLT-based transcoding performs rather poor. Therefore we are investigating optimization for the transcoding. Caching of transcoded contents is one conceivable approach. Another one may be to use compiled XSLT style sheets. This may lead to a significant speed-up and preserve the advantages of XSLT transcoding, such as flexibility. In the future we plan to inte-

grate additional adaptation filters for other media into the adaptation framework. Due to the extensible architecture of the adaptation framework, this should be easily done.

ACKNOWLEDGEMENT

The authors want to acknowledge the contribution by Sebastian Weber, Gregor Franke, and Gerald Hübsch to the design of DDL and the implementation of the adaptation framework.

REFERENCES

[1] Abrams, M., Helms, J., 2000. User Interface Markup Language (UIML) v3.0 Draft Specification, Harmonia, Inc., (http://www.uiml.org/).

[2] Bickmore, T., Girgensohn, A., Sullivan, J. W., 1999. Web Page Filtering and Re-Authoring for Mobile Users. *The Computer Journal*, Vol. 42, No. 6, pp. 534-546.

[3] Biron, P., Malhotra, A., 2001. XML Schema Part 2: Datatypes. W3C Recommendation. (http://www.w3.org/-TR/xmlschema-2/).

[4] Buyukkokten, O., Garcia-Molina, H., Paepcke, A., 2001. Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices. *Proc. of 10th Int. World Wide Web Conference.* Hong Kong, pp. 652-662.

[5] Dubinko et al, "XForms 1.0", W3C Working Draft, 2001. (http://www.w3.org/TR/xforms/).

[6] Fox, A, Brewer, E., 1996. Reducing WWW latency and bandwidth requirements by real-time distillation. *Fifth International World Wide Web Conference.* Paris, France.

[7] Fox, A., Goldberg, I., Gribble, S.D., Polito, A., Lee, D.C., 1998. Experience with Top Gun Wingman: A proxy-based graphical web browser for the Palm Pilot PDA. *IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98).* Lake District, UK, pp. 15-18.

[8] Fox, A., Gribble, S. D., Chawathe, Y., Brewer, E. A., 1998. Adapting to Network and Client Variation Using Active Proxies: Lessons and Perspectives. *IEEE Personal Communications*, August 1998. pp. 10–19.

[9] Goebel, S. et al, 2001. Device Independent Representation of Web-based Dialogs and Contents. *Proceedings of IEEE Youth Forum in Computer Science and Engineering (YUFORIC'01).* Valencia, Spain, pp. 69-76.

[10] Ito, N., Manabe, K., 2000. XML Document Navigation Language. W3C Note by NEC Corporation.

[11] Klyne, G. et al, 2001. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies. W3C Working Draft. (http://www.w3.org/TR/CCPP-struct-vocab/).

[12] Muffin Web-Proxy Homepage (http://muffin.doit.org).

[13] Raggett, et al, 1999. HTML 4.01 Specification, W3C Recommendation. (http://www.w3.org/TR/html4/).

[14] Schilit, B.N., Trevor, J., Hilbert, D., Koh, T.K., 2001. m-Links: An Infrastructure for Very Small Internet Devices. *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking.* Rome, Italy, pp. 122-131.

[15] WAP Forum Ltd, 2000. Wireless Application Protocol, Wireless Markup Language Specification, Version 1.3. http://www.wapforum.org/what/technical.htm.

[16] Watson, T., 1994. Application Design for Wireless Computing. *Proceedings of the 1st IEEE Workshop on Mobile Computing Systems and Applications (MCSA '94), Santa Cruz, California.* pp. 91-94.

[17] IBM WebSphere Transcoding Publisher, IBM Corp. http://www-4.ibm.com/software/webservers/transcoding/.

[18] What's i-mode?, NTT DoCoMo,Inc. http://www.nttdocomo.com/i/service/home.html.

[19] Xalan, Java Version 2.2 D9, The Apache Software Foundation. http://xml.apache.org/xalan-j.

[20] Xerces Java Parser 1.2.3, The Apache Software Foundation. http://xml.apache.org/xerces-j.

[21] Zenel, B., 1999. A general purpose proxy filtering mechanism applied to the mobile environment. *Wireless Networks of the ACM*. Vol. 5, No. 5, pp. 391-409.