AMACONT: A System Architecture for Adaptive Multimedia Web Applications

Michael Hinz, Zoltán Fiala

Dresden University of Technology
Heinz-Nixdorf Endowed Chair for Multimedia Technology
Mommsenstr. 24, D-01062 Dresden
{michael.hinz, zoltan.fiala}@inf.tu-dresden.de

Abstract: Engineering personalized ubiquitous Web applications requires to develop adaptable Web content as well as to automatically adjust it to varying client devices and dynamically changing user preferences. To meet this requirement, this paper introduces the system architecture of the AMACONT project [Am04] that aims at the dynamic generation of Web presentations tailored to users' varying needs and device capabilities. Different techniques for static and dynamic content, layout and structure adaptation supported by the overall architecture are explained. Furthermore, strategies for dynamic device and user modeling are illustrated in order to guarantee up-to-date user and device models for the whole content generation process. Finally, modules of a modular graphical authoring tool for the visual development of adaptive Web applications are presented.

1 Introduction

Providing personalized information becomes an important challenge of today's Web development. The raising number of users with an increasing variety of mobile devices requires the creation and publication of content that is tailored to users' needs and platform capabilities. Effective reuse mechanisms have to be evolved for presenting information or parts of it in varying contexts. Furthermore, a strict separation of different concerns like content, navigation and presentation is required in order to store structure and information in a device and user independent way. Finally, adaptation rules referencing user models and device profiles must be added to the presentation and dynamically processed at run-time for each user request.
Recently, different approaches for adapting content to different contexts have emerged. NAC (Negotiation and Adaptation Core) is a basic core for multimedia services adaptation and negotiation in heterogeneous environments [LL01]. However, since this approach necessitates special modules and browser extensions on the client side, it is not practical for straightforward use on arbitrary clients. In [LL02] a decision engine with QoS awareness is proposed that can automatically negotiate the appropriate adaptation or transcoding strategies for producing an optimal adapted version. Several transcoding methodologies employed by the IBM WebSphere Transcoding Publisher product are described in [Br01]. Nevertheless, as transcoding techniques only transform existing Web content like HTML to other representations, these approaches are limited.

Other approaches focus on content adaptation in specific application domains and are therefore limited to those scenarios. Most of them (e.g. [ZPS01], [Gr97]) concentrate on adaptation in the location based service or the local information domain. In the domain of virtual communities the Community-Driven Adaptation [Mo04] middleware architecture is used to adapt content based on user feedback.
In this paper a system architecture for the development and delivery of general-purpose adaptive cross-media Web applications is presented. Based on the concept of reusable Web document components it provides a framework for the dynamic transformation of content to a dynamically changing user and device context.

2 A System Architecture for Adaptive Web Content

The main goal of our architecture is the generation of personalized applications for the ubiquitous Web. This assumes to provide adaptive intelligent user interfaces addressing heterogeneous capabilities of device classes and an automatic adjustment of content regarding those device classes. The overall architecture of AMACONT was realized on the basis of the XML publishing framework Cocoon [Co04]. It aims at transforming adaptive Web components to Web pages adapted to user properties and preferences as well as device profiles. This process is performed in a stepwise, pipeline oriented way (see Figure 1).

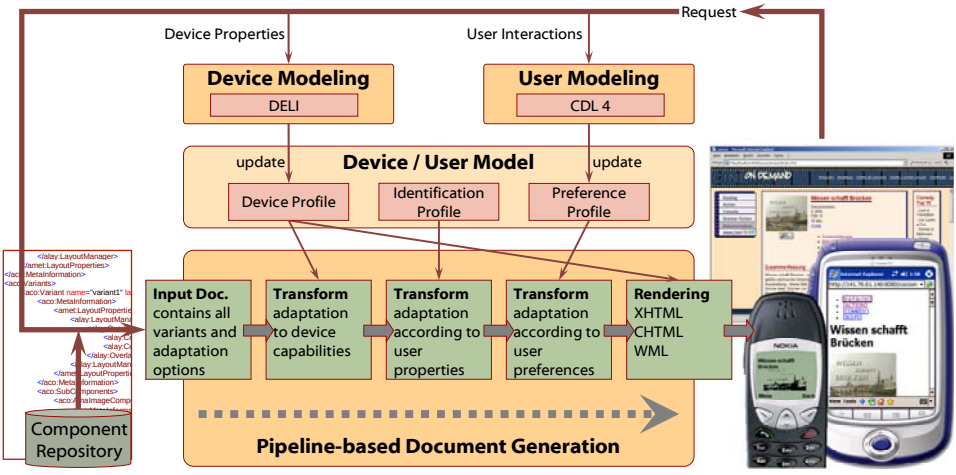

Figure 1: Dynamic Generation Process of Personalized and Device Dependent Web Pages

For each user request, a complex document component encapsulating all possibilities concerning its content, layout, and structure is retrieved from a component repository. A short introduction to the underlying component-based document format and its support for adaptation is given in Section 2.1. According to the device and user model, this

document is subdued to a series of XSLT and JAVA transformations, each considering a certain adaptation aspect by the configuration and selection of components (see section 2.2). After the resulting (adapted) component hierarchy is determined, it has to be presented in a specific output format (XHTML, cHTML, WML etc.). According to the layout managers described in Section 2.2.3, this rendering happens automatically.
To make sure that the document generation is always based on an up-to-date user and device model, these models have to be permanently updated according to the user's browsing behavior. A detailed introduction to the dynamic user and device modeling techniques is given in Section 2.3.

2.1 Document Model

The component-based document format of AMACONT aims at building personalized ubiquitous Web applications by aggregating and linking configurable document components. These components are instances of an XML grammar representing multimedia Web content on different abstraction levels. *Media components* encapsulate concrete media assets (e.g. text, image, video and audio) by describing them with technical metadata. *Content units* group media components by declaring their layout in a device-independent way. Finally, *document components* define a hierarchy out of content units to fulfill a specific semantic role. The *hyperlink level* for defining typed links is spanned over all component layers. The main benefits of the document model are the support for reuse of content via component technology and the support for user and device adaptation of dynamic generated Web pages. Therefore the model is suitable for cross media applications displaying information in varying context. For a detailed introduction to the document model the reader is referred to [Fi03].

2.2 Adaptation during Pipeline-based Generation

As mentioned above, during the document generation process an input document containing adaptation rules is adjusted to a dynamic device and user model. Figure 1 shows a possible scenario containing three steps. The first two adaptation steps are performed according to the variant selection mechanism described in section 2.2.1. This way the hierarchy of components is adjusted to static user properties (age, gender, knowledge level, etc.) as well as device profiles and classes (e.g. PDA, cell phone or notebook). The third step adjusts the media components contained in this transient document to changing user preferences by varying their occurrence. Finally, the resulting document is transformed to the requested output format (e.g. XHTML, cHTML, WML).

2.2.1 Adaptation in Dependency of User and Device Properties

To define adaptive behavior in a generic way, each component may include a number of variants. As an example, the definition of an image component might include (in its

body) two alternatives for color and monochrome displays. Similarly, the number, structure, arrangement, and linking of subcomponents within a document component can also vary depending on device capabilities or user preferences. The decision which alternative is selected is made during document generation by a Java-based transformer according to a selection method which is declaratively described in the component's header. Such selection methods are chosen by component developers at authoring time and can represent arbitrarily complex conditional expressions parameterized by user and device model parameters. The XML-grammar for selection methods allows the declaration of user model parameters, constants, variables, and operators, as well as complex conditional expressions (such as if or case) of arbitrary complexity [Fi03]. Note that alternatives can be declared for components of all granularities, thus allowing to define adaptive behavior on different abstraction levels.

2.2.2 Adaptation in Dependency of Rules Representing User Preferences

While the mechanism described in Section 2.2.1 takes the whole component hierarchy of the Web presentation into account, another adaptation technique for dynamically changing the appearance of single media components has been developed. Aim of this technique is to allow users to interact on media components, to automatically derive user preferences from those interactions and to dynamically update the resulting Web presentation according to these preferences. Note that this strategy can be effectively used for optimizing Web pages on mobile devices with limited presentation space. Figure 2 shows an example of an interactive multimedia Web presentation allowing to perform interactions on an image object. A user being more interested in textual information (due to the limited display capabilities of his browser) could collapse images and enlarge texts. According to this interaction, the system can update the user model and generate the following Web pages according to this new model.

Figure 2: Interactive Multimedia Web Presentation: By collapsing the picture more space for other presentation objects is available

For representing and dynamically updating such user preferences CDL4 [Sh96], an incremental learning algorithm, is used. It utilizes a dynamic set of decision rules for expressing user interests and refines those rules in a stepwise way according to new user interactions. During document generation, the appropriate rules are evaluated and the occurrence of the affected media objects is adjusted. For a detailed description of this mechanism the reader is referred to [HFW04]. In this paper the acquirement of user interactions and their processing within the architecture is detailed (see Section 2.3.2).

2.2.3 Automatic Layout Adaptation

After the component hierarchy to be presented and the parameters of media objects have been determined, the resulting adapted document has to be transformed to a specific presentation format (XHTML, cHTML, WML etc.). This happens automatically according to the AMACONT document format, which allows attaching XML-based layout descriptions to components. Inspired by the layout manager mechanism of the Java language (AWT and Swing), they describe a client-independent layout allow abstracting from the exact resolution of the display or the browser's window. Note that layout managers of a given component only describe the presentation of its immediate subcomponents, which encapsulate their own layout information in a component-based way. At current time four layout managers: *BoxLayout*, *BorderLayout*, *GridTableLayout* and *OverlayLayout* can be defined. The exact adjustment of media objects according to the layout managers happens during document generation time by XSLT stylesheets that transform components with such abstract layout properties to specific output formats.

2.3 Device and User Modeling

Having an always up-to-date user and device model is crucial for generating personalized ubiquitous Web pages. This necessitates the acquisition of existing device constraints, user properties and user preferences.

2.3.1 Acquiring device properties

To acquire device properties several strategies exists. The most popular method is to analyze the HTTP user agent parameter and map the result to a device or browser repository on the server side. This works only for nearly static properties because of the small and not standardized vocabulary of the user agent. Hence, the CC/PP (Composite Capability / Preference Profiles) standard was established, an RDF grammar for describing device capabilities and user preferences in a standardized way [Kl04]. However, as being a general grammar, CC/PP makes no assumptions on concrete resource characteristics.

Therefore, our system architecture uses the WAP User Agent Profile (UAProf [Wi01]) providing a common vocabulary for WAP devices.
To support also other mobile devices (e.g. PDAs), specific extensions of UAProf have been made. This extended format is handled by DELI [Bu02] on the server side. DELI provides an API to allow Java servlets to determine the capabilities of a client device using CC/PP or UAProf. Whereas today nearly only WAP devices support UAProf, our system has to autonomously collect the devices properties of other end devices (e.g. Notebook, PDA) via client side scripts. Those scripts are automatically included in the presentation during document generation. Every time the user requests a new document, those scripts send the gathered properties with the corresponding HTTP-Request to the server. The server processes that information through a device modeling component by forming a new profile or updating an existing profile with profile differences (*profile-diff*), which are calculated according to the collected client properties (where only the changed properties are included). To sum up, this mechanism gives the chance for an effective and unitized handling of device capabilities of different device classes. Furthermore, it enables to acquire permanently changing device properties (e.g. size of the browsers window) by collecting that information via scripts directly on the client device. The result is an always up-to-date device profile in the device/user model on which the document generation is based on.

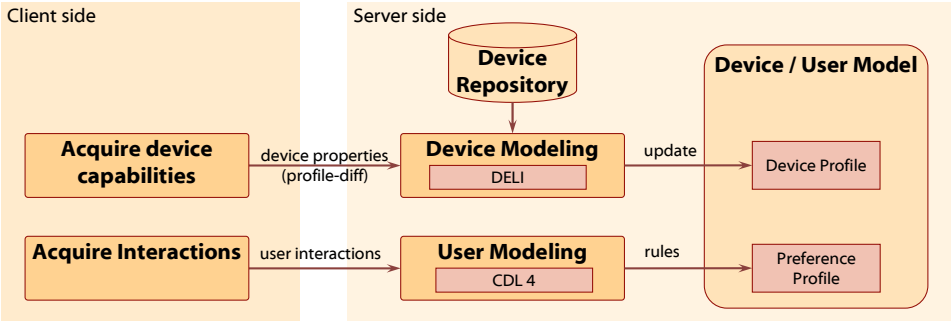

Figure 3: Device and User Modeling Process

2.3.2 Modeling User Preferences

As mentioned in Section 2.2.2, adapting content to dynamic user preferences can be effectively used for optimizing Web pages on mobile devices. Other approaches like [Go01], [Pa04] or [Ho00] only clip or restructure Web pages to make them suitable for limited mobile devices. However, capturing such preferences is a serious problem if we do not want to explicitly ask the user to give explicit information about his/her preferences. Our developed system allows observing users' browsing behavior by tracking interactions being performed on media components. Similar to the acquirement of device capabilities this is done by means of specific code fragments (JavaScript or JScript) which are embedded and configured for each media component during document gen-

eration. They allow capturing user interactions on the client side and sending them back to the server, where they are stored in history lists (session profile). In this way several media objects (e.g. video, audio, image, text) of the Web page can be observed in order to acquire interactions with them (e.g. video started/paused/stopped, image minimized/maximized, image printed, text enlarged/collapsed, text scrolled).

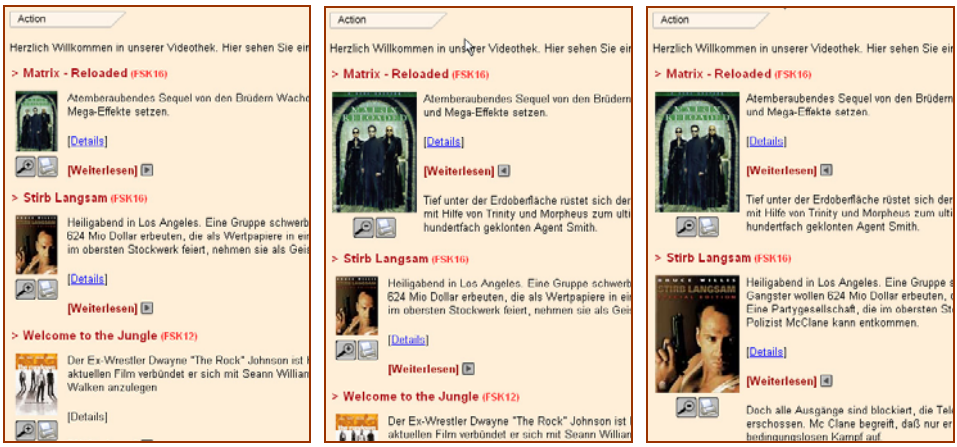

Figure 4: Automatic User Modeling by Observing and Processing User Interactions

Figure 4 shows a possible sequence of automatically generated XHTML documents of an online video store. In the left picture the user enters the default version of the page containing no detailed information. As mentioned in Section 2.2.2, the user's preferences are represented in form of a decision list. In the beginning of a user session, the following "trivial" rule is created:

[(default → noInterest)]

Note that an XML grammar has been developed for representing such rules. Still, for the sake of readability we use the above simplified formalization in this paper.
When the user is interested in getting enhanced information about videos, she maximizes the title picture or enlarges a more detailed text description (see Figure 4, middle). This interaction is captured by client-side scripts and sent to the server in the following form:

```
<component id="movie_picture" type="ImageComponent">
        <userEvent name="maximized" time="1088776213776" /> </component>
<component id="movie_text" type="TextComponent">
        <userEvent name="maximized" time="1088776246233" /> </component>
```

Based on this interaction the CDL4 algorithm is triggered, which adds the corresponding rules to the user model:

[((medium ≠ picture) Λ (medium ≠ text) → noInterest),
((category ≠ action) → noInterest),
(default → interest)]

When the user comes back to this Web page or any other page containing the video list, an adapted presentation according to the updated rules is generated (see Figure 4, right).
In order to establish the connection between low-level interactions (e.g. enlarging pictures) and the rule semantics (interest in action films), component authors have to provide the observable media components with specific metadata. In the above example, semantic metadata in the form of attribute-value pairs (e.g. category = ”action”) is attached to the affected image component and evaluated by the CDL4 module.

3 Authoring Tools

In order to visually develop AMACONT applications a modular authoring tool is being developed. It utilizes a flexible object model based on JDOM to provide programmatic access to AMACONT documents. Furthermore, it allows to associate different types of custom editor plugins with arbitrary AMACONT components. Thus, different graphical editor modules for supporting selected steps of the overall authoring process can be developed and “plugged in” in the authoring framework.
Recently, basic editors for creating adaptable media components (text, image, video, audio, CSS) have been created. As an example for such media editors we mention the image editor allowing to upload images in different formats (e.g. jpeg, gif, bmp, png), to edit their properties in a visual way and to save them as image components. The size of an image can be altered either by mouse dragging or by determining the explicit pixel values by input fields. Furthermore, mechanisms for specifying the adaptive behavior of image components have been integrated, too. Firstly, it is possible to provide an alternative text for browsers that are not able to present images. Secondly, image variants with different quality alternatives can be added. Such variants can be created in three ways: by uploading alternative images, by resizing the current image and save it as a new variant or by generating new images automatically. In the latter case the author can predefine the properties (e.g. pixel size, color depth, image format) of an arbitrary number of variants to be created. According to this configuration, both the new media instances and the corresponding decision logic (see Section 2.2.1) are generated automatically. This feature was implemented using the Java API of ImageMagick [Im04].
Another existing module is the layout editor shown in Figure 5 that aims at specifying the device independent layout of component based Web documents. It allows to attach layout managers (see Section 2.2.3) to components and to configure the attributes of those layout managers in an in-tuitive visual way. Layout managers are visualized by means of a grid that can be filled by icons representing subcomponents. Various “Drag&Drop” techniques have been realized in order to perform most operations graphically, such as resizing the grid, placing subcomponents into grid cells, changing their alignment etc. Besides, various input fields for fine-tuning all possible layout attributes can be found on the right editor pane. Furthermore, a preview function for testing the current layout in XHTML has been developed, too.

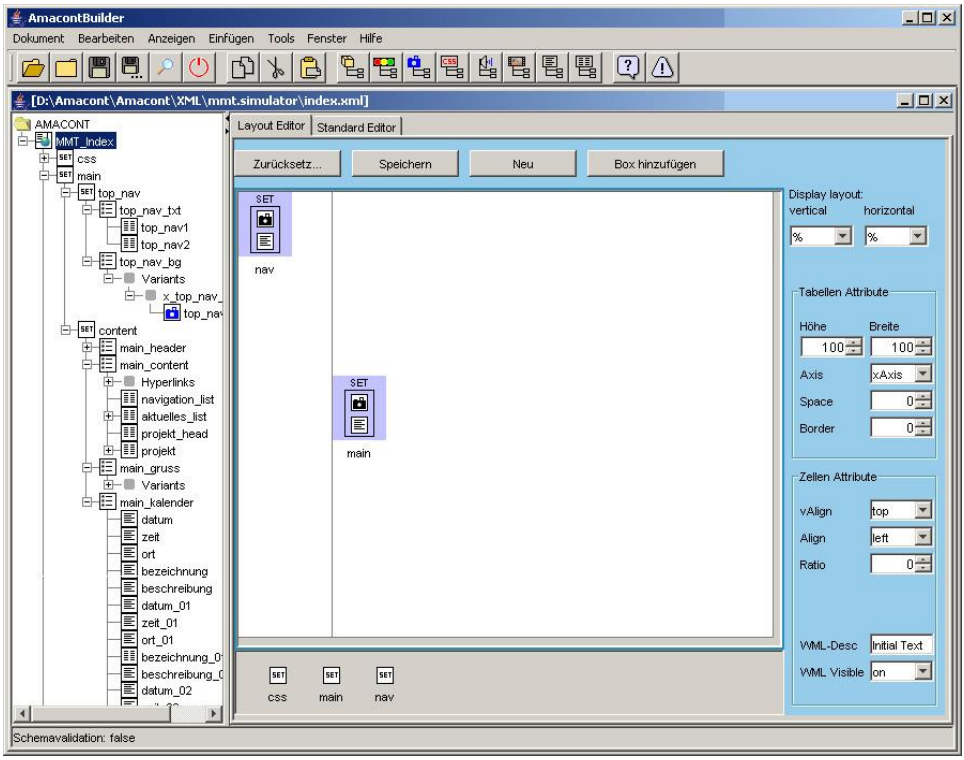

Figure 5: Layout Editor Tool for Specifying Device Independent Layout

4 Conclusion and Future Work

In this paper an overview of the AMACONT system architecture for the generation of personalized ubiquitous applications was given. Different mechanisms for adjusting reusable Web content to different user preferences and device capabilities, as well as techniques for keeping user and device models up-to-date were shown. Finally, first steps towards a visual authoring framework for component-based adaptive Web applications were presented.

Ongoing work concentrates on optimizing the performance of the architecture and reducing the server load when handling a multiplicity of users. Initial tests showed that especially dynamic adaptation mechanisms cause heavy server load. In future work we will show how the distribution of parts of the overall architecture could improve the overall performance. Besides distributing server components on different servers the outsourcing of server code to the client devices will be examined, too. Furthermore, additional modules of the authoring tool for specifying adaptation issues will be developed.

References

[Am04] AMACONT Project Homepage
http://www-mmt.inf.tu-dresden.de/english/Projekte/AMACONT/

[Bu02] MarkButler, M.: DELI: A DElivery context LIbrary for CC/PP and UAProf. External Technical Report HPL-2001-260 (revised version 02/08/2002), 2002.

[Br01] Britton, K. H.; Case, R.; Citron, A.; Floyd, R.; Li, Y.; Seekamp, C.; Topol, B.; Tracey, K.: Transcoding: Extending e-business to new environments. In IBM Systems Journal, Technology for e-business, Volume 40, Number 1, 2001.

[Co04] The Apache Cocoon Project Homepage, http://cocoon.apache.org/

[Fi03] Fiala, Z., Hinz, M., Meissner, K., Wehner, F.: A Component-based Approach for Adaptive, Dynamic Web Documents. In Journal of Web Engineering, Rinton Press, Vol.2 No.1&2, (pp058-073), September 2003.

[Go01] Gomes, P.; Tostao, S.; Goncalives, D.; Jorge, J.: Web Clipping: Compression Heuristics for Displaying Text on a PDA. In Proceedings of the Mobile HCI'01, 2001.

[Gr97] Abowd, G.; Atkeson, C.; Hong, J.; Long, S.; Kooper, R.; Pinkerton, M.: Cyberguide: A Mobile Context-Aware Tour Guide. In Proceedings of the ACM Wireless Networks conference, pages 421–433, 1997.

[HFW04] Hinz, M.; Fiala, Z.; Wehner, F.: Personalization-based Optimization of Web Interfaces for Mobile Devices. In Proceedings of the MobileHCI 2004, Glasgow, Scotland, September 2004.

[Ho00] Hori, M.; Kondoh, G.; Ono, K.; Hirose, S.; Singhai, S.: Annotation-based Web Content Transcoding. In Proceedings of the World Wide Web 9 conference, Amsterdam, Netherlands, 2000.

[Im04] ImageMagick hompage:
http://www.imagemagick.org/

[Kl04] Klyne, G.; Reynolds, F.; Woodrow, C.; Ohto, H.; Hjelm, J.; Butler, M.; Tran, L.: Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation 15 January 2004.
http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/

[LL01] Lemlouma, T.; Layaïda, N.: NAC: A Basic Core for the Adaptation and Negotiation of Multimedia Services, Opera Project, Inria, September 2001.

[LL02] Lum, W. Y.; Lau, F. C. M.: A Context-Aware Decision Engine for Content Adaptation. In the IEEE Pervasive Computing journal, July-September 2002 (Vol. 1, No. 3), pages 41-49.

[Mo04] Mohammed I.; Chin A.; Cai J.; De Lara E.: Community Driven Adaptation: A Middleware Architechture for Automatic Content Adaptation in Pervasive Enviroments. In Proceedings of the International Middleware 2004 conference, 2004.

[Pa04] Palm Web Clipping Resources:
http://www.palmos.com/dev/tech/webclipping/resources.html

[Sh96] Shen, W.: An efficient Algorithm for Incremental Learning of Decision Lists. Technical Report, USC-ISI-96-012, University of Southern California, 1996.

[Wi01] Wireless Application Group: User Agent Profile Specification. Open Mobile Alliance WAP Forum 2001.

[ZPS01] Zarikas, V.; Papatzanis, G.; Stephanidis, C.: An architecture for a self-adapting information system for tourists. In Proceedings of the Workshop on Multiple User Interfaces over the Internet, 2001.