An XML-based Component Architecture for Personalized Adaptive Web Applications

Zoltán Fiala, Michael Hinz, Frank Wehner

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

Abstract: Developing personalized applications for the ubiquitous Web assumes to create content that can be automatically adapted to both different presentation platforms and user preferences. To answer this need, the project AMACONT [Am03] recently introduced a component-based XML document format. It enables to compose personalized Web applications by the aggregation and linkage of fine-granular document components from different abstraction levels. This paper aims at a detailed description of personalization issues regarding both content and device adaptation. XML technologies are used to express device profiles and user models, as well as to define adaptive behavior and layout of components in a generic way. Finally, the developed pipeline-based document generator for dynamically transforming adaptable component structures to different Web output formats is explained.

1 Introduction

Today's Web is quickly evolving towards an interactive medium that offers up-to-date information for a rapidly growing audience characterized by a large diversity of interests and client devices. However, existing document formats (such as HTML, cHTML or WML) are hardly suitable for engineering personalized ubiquitous Web applications. The lacking separation of content, layout and structure prevents to uniformly create and update content for different preferences and platforms. Furthermore, no mechanisms for describing the adaptive behavior of content pieces in a generic way are provided.
Recently, different approaches for modeling and engineering adaptive hypermedia and Web systems have emerged. Reference models aiming at finding common abstractions to hypermedia systems, like Dexter [HS94] or the model of Tochtermann [To94] have been extended towards adaptation e.g. by AHAM [DHW99] or the Munich Reference Model [KW02]. Significant research has been done on design and process models covering hypermedia development phases, too. Approved design principles like OOHDM [SRB96] or RMM [ISB95] have been enhanced with personalization issues in [RSG01] or by Hera [FHV02]. However, all these approaches focus on the conceptual modeling and design of hypermedia applications, not supporting the flexible reuse of adaptable implementation artefacts.

There exist only a few approaches towards reusing implementation entities in hypermedia development. The WebComposition Markup Language [GSG00] enables the component-based development of Web applications. Westbomke [WD02] proposes a formal XML-grammar for the implementation and presentation of platform-independent structured hypermedia documents by formalizing the concepts of Tochtermann's model. Still, adaptation to device capabilities and changing user preferences is not a central aspect of these approaches.
In [WS01] an approach called „Intensional Hypertext“ is introduced. The „Intensional Markup Language“ (IML) is a proprietary extension of HTML containing control structures for describing adaptive behavior on the basis of the chosen URL and user input. However, there is no support for output formats other than HTML.
To fill this gap, the project AMACONT [Am03] recently introduced a component-based document format for personalized ubiquitous Web presentations [Fi03]. It focuses not on the conceptual design of Web applications, but on the challenge to reuse adaptable implementation artefacts. In this paper a detailed overview of personalization issues is given. XML technologies are used to describe device capabilities and user preferences, as well as to define the adaptive behavior and layout of components. Finally, the developed pipeline-based document generator is described.

2 The Document Model

The component-based document format [Fi03] allows to build device-independent Web applications by aggregation and linkage of configurable document components. These are documents or document fragments, instances of an XML-grammar representing adaptable content on different abstraction levels (see Figure 1). Components' interfaces are described by metadata specifying their properties and adaptive behavior. The format was defined by XML Schema.
The lowest level introduces *media components* which encapsulate concrete media assets. These comprise text, structured text (e.g. HTML), images, sound, video, Java applets and may be extended arbitrarily. Beside technical properties expressed by MPEG-7 descriptors, additional content management information is provided, too.
On the second level media components belonging together semantically - e.g. an image with textual description – are combined to so called *content units*. Defining such collections is a key factor of reuse. The spatial adjustment of contained media components is described by client-independent layout properties abstracting from the exact resolution and presentation style of the current display (see Section 3.2).
Thirdly, *document components* are specified as parts of Web presentations playing a well defined semantic role (e.g. a news column, a product presentation or even a Web site). They can either reference content units, or aggregate other document components. The resulting hierarchy describing the logical structure of a Web site is strongly dependent from the application context. Again, the spatial adjustment of subcomponents is described in a client-independent way.

Finally, the orthogonal *hyperlink view* defines links spanned over all component levels. Uni- and bidirectional typed hyperlinks based on the standards XLink, XPath and XPointer are supported.

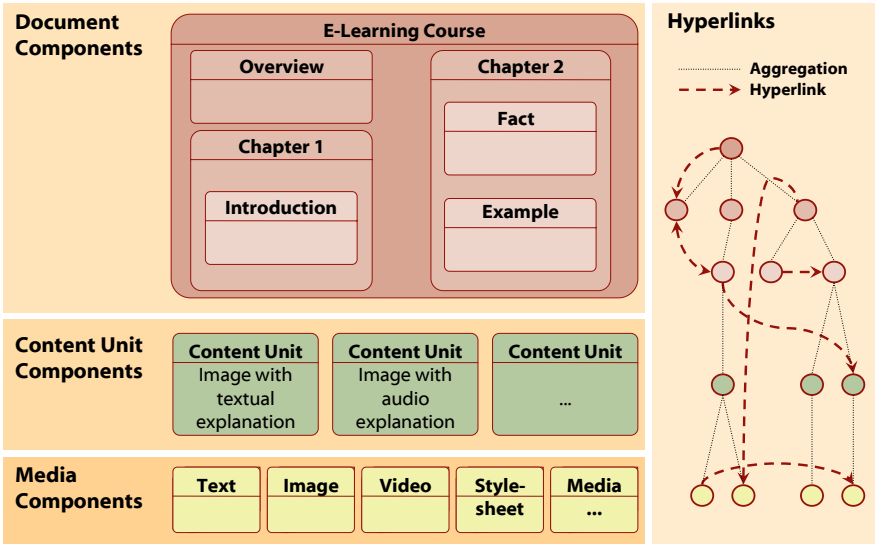


Figure 1: The Document Model

3 Personalization Issues

The document format supports personalization by encapsulating adaptive behavior in components on different abstraction levels. Firstly, adaptation is required on the level of media components in order to consider various client capabilities or other technical preferences (e.g. bandwidth, color depth, etc.) by providing alternative media instances with varying quality. Secondly, on the level of content units the number, type and arrangement of inserted media components can be adjusted. Consider the case of two online-shop customers, one of them preferring detailed textual descriptions, the other visual information. The presentation for the first user might include content units containing text objects, for the other one rather images or videos. Thirdly, personalization of document components concerns the adaptation of the whole component hierarchy, which results in different subcomponent trees for different user preferences and/or device capabilities. Finally, adapting hyperlinks enables personalized navigation structures within the generated Web presentation.

3.1 Describing Adaptive Behavior

In order to describe 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 two variations 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 an XSLT stylesheet according to a certain selection method which is described in the component's header. Such selection methods are chosen by component developers at authoring time and can represent arbitrary complex conditional expressions parameterized by user model parameters. This separation of describing variants (in the component body) and adaptation logic (in the component header) allows reusing a given component in different adaptation scenarios. The XML code below demonstrates the definition of a document component's variants and a selection method. In a Web presentation offering TV programs, different content depending on the user's age is presented.

```
<AmaDocumentComponent name="TVProgram">
    <MetaInformation>
        ...
    </MetaInformation>
    <Variants>
        <Variant name="Adult_Program">
            ...
        </Variant>
        <Variant name="Child_Program">
            ...
        </Variant>
    </Variants>
</AmaDocumentComponent>
```

```
<AdaptiveProperties>
  <If>
    <Expr operator="greaterThan">
        <UserParam>
          UserAge
        </UserParam>
      <Const>18</Const>
    </Expr>
    <Then res="Adult_Program"/>
    <Else  res="Child_Program"/>
  </If>
<AdaptiveProperties>
```

Table 1: Defining component variants (left) and selection methods (right)

The processing XSLT style sheet substitutes the integer variable "User.Age" by its value from the current user model (see Section 3.3), performs the selection method and determines the proper variant of the "TVProgram" component. As this variant might also have varying subcomponents, the style sheet works recursively. The XML-grammar for selection methods allows the declaration of user model parameters, constants, variables and operators, as well as complex conditional expressions of arbitrary depth. The processing XSLT stylesheets acts as an interpreter for this "selection method language".

3.2 Automatic Layout Adaptation

Defining selection rules for component variants is an effective tool for content and link adaptation, but it requires to involve the author. A different issue is the visual adjustment

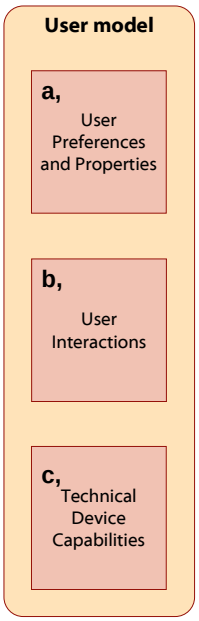
of component hierarchies according to graphical capabilities of varying end devices. This can be mostly done automatically, without additional authoring support.
As described in Section 2, the document format enables to describe the spatial adjustment of subcomponents within their container components by client-independent layout properties. Inspired by the layout manager mechanism of the Java language (AWT and Swing), these properties describe a size- and client-independent layout allowing to abstract from the exact resolution of the display or the browser's window. The exact rendering of media objects is done by XSLT stylesheets transforming these abstract layout descriptions into concrete output formats. At current time four layout managers: *BorderLayout*, *BoxLayout*, *OverlayLayout* and *GridLayout* can be defined, and three XSLT stylesheets for XHTML, cHTML and WML output were realized. Figure 2 presents a component's XML code containing abstract layout properties (marked gray), as well as the result of the automatic rendering process. Note that layout managers of a component reference its subcomponents, which may also have variants according to certain adaptation aspects. The combination of author-driven content adjustment and automatic layout adaptation techniques in the document generation process will be described in Section 4.

Figure 2: Adapting Layout to Different Device Formats

3.3 The User Model

The adaptation of components during document generation happens according to an XML-based user model. This contains information on user preferences and properties, user interactions and technical client capabilities (see Figure 3).

Each part of the user model relies on CC/PP (Composite Capability/Preference Profiles), an RDF grammar for describing device capabilities and user preferences in a standardized way [Kl02]. However, as being a general grammar, CC/PP makes no assumptions on concrete resource characteristics. Though a standard called WAP User Agent Profile (UAProf [Wi01]) providing a common vocabulary for WAP devices has been defined on the basis of CC/PP recently, it can not sufficiently describe alternative devices, like e.g. PDAs or TabletPCs. Thus, in order to support a larger diversity of end devices a new profile in the style of UAProf was defined and integrated in the user model (Figure 3c).




```
<UserModel>
    <Description ID="UserProfile">
        <Description ID="UserProperties">
            ...
            <Description ID="General">
                <Name>Steven Test</Name>
                <UserAge>21</UserAge>
                ...
        </Description>
        <Description ID="UserInteraction">
            <Events>
                ...
                <component id="MainText" type="TogTextComp">
                    <userEvent name="printed" time="0304162030"/>
                    ...
        </Description>
    </Description>
    <Description ID="DeviceProfile">
        <Description ID="SoftwarePlatform">...</Description>
        <Description ID="HardwarePlatform">...</Description>
        <Description ID="NetworkCharacteristics">
            …  <AverageBandwith>54000</ AverageBandwith > …
    </Description>
</UserModel>
```

Figure 3: User Model Structure

In addition, another CC/PP grammar for describing user profiles in a generic way was developed. Firstly, it allows to declare user preferences and properties (Figure 3a). Beside a set of general user properties, arbitrary extensions are allowed. Secondly, the grammar also enables to store past user interactions in the form of *events* related to components (Figure 3b). As an example, the corresponding XML fragment above describes that the user has printed out some text represented by the component called “MainText” via his Web browser.

History lists describing user interactions provide a useful mechanism for implementing dynamic personalization, i.e. personalization within the generated Web presentation according to changing user preferences. By evaluating interactions (e.g. visiting pages, following links, starting videos, interrupting downloads, etc.) suggestions on the current user’s preferences and knowledge can be made and parts of the user model describing user preferences can be updated or specialized (see the upper region of Figure 4). In a developed prototype application for product presentation this specialization is performed

by the incremental learning algorithm CDL4 [Sh96]. The algorithm was approved as very useful in adaptive multimedia product presentations in an earlier project of the authors' research group [JM98].
In order to acquire most user model parameters automatically (i.e. without explicitly asking the user), client-side mechanisms based on JavaScript routines or MIDlets (for J2ME capable devices) were developed. These allow to gather information on client capabilities and user interactions, as well as to send it to the server according to the above mentioned CC/PP-based profiles. These client-side "watching" routines are automatically inserted and configured at document generation.

4 The Document Generator

Document generation is based on a stepwise pipeline concept (Figure 4). For each user request, a complex document encapsulating all possibilities concerning its content, layout, and structure is retrieved from a component repository. According to the user model, it is subdued to a series of XSLT transforms, each considering a certain adaptation aspect by the configuration and selection of component variants (see Section 3.1).

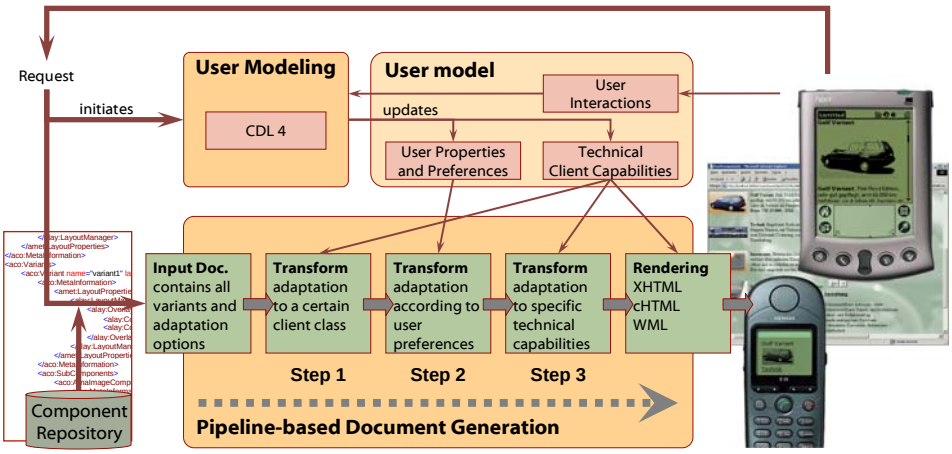


Figure 4: Pipeline-based Document Generation

Instead of evaluating all adaptation rules "at once", the generation process can be divided into more steps in order to reuse partially adapted documents for similar requests.
Figure 4 shows a possible scenario with three steps, namely adaptation to a certain client class (e.g. PDA, cell phone or desktop browser), then to semantic user preferences (interests, knowledge level, media preferences, etc.) and finally to specific technical capabilities (e.g. bandwidth, display resolution). Though the three XSLT stylesheets are identical, they are parameterized differently, so that each of them processes only specific decision rules. As an example, the first one evaluates only rules referencing variables describing the client class and leaves other rules unprocessed. As an example, when the

same document is requested by two PDA users, the output of this transform can be reused, even if those users have different preferences concerning their interests or knowledge level.
After all adaptation rules have been evaluated and the final static component hierarchy – without variants – has been determined, a Web document in a specific output format (XHTML, cHTML, WML etc.) is generated. The concrete rendering of components happens automatically (see Section 3.2). The document generator was realized based on the publishing framework Cocoon [Co03].

5 Conclusion and Future Work

In this paper a detailed overview of personalization and adaptation issues provided by the XML-based component architecture of the project AMACONT [Am03] was given. The component-based document format supports the flexible reuse of fine-granular adaptive Web implementation units. It provides generic mechanisms to describe the adaptive behavior of components on different abstraction levels as well as their adaptive layout in a client-independent way. The modular document generator architecture enables the stepwise transformation of component structures to different output formats according to an XML-based user model.
The concepts described in this paper have been prototypically implemented in a prototype for product presentations. Ongoing work concentrates on the authoring process of component-based Web applications. Existing design and process models for hypermedia application development are analyzed regarding their extensibility for adaptive component-based Web sites. Furthermore, a component repository and a modular authoring tool for creating, storing, retrieving, configuring, and publishing adaptive content components will be designed and implemented.

References

[Am03] AMACONT Project Homepage
http://www-mmt.inf.tu-dresden.de/english/Projekte/AMACONT/

[Co03] The Apache Cocoon Project Homepage, http://cocoon.apache.org/

[DHW99] De Bra, P., Houben, G., Wu, H.: “AHAM: A Dexter-based Reference Model for Adaptive Hypermedia”, Hypertext ’99, Darmstadt (1999)

[Fi03] Fiala, Z., Hinz, M., Meißner, K., Wehner, F.: “A Component-based Approach for Adaptive, Dynamic Web Documents”; WWW2003, Budapest 2003

[FHV02] Frasincar, F., Houben, G., Vdovjak, R.: “Specification Framework for Engineering Adaptive Web Applications”, WWW11, 2002

[GSG00] Gaedke, M., Segor, C., Gellersen, H.-W.: “WCML: Paving the Way for Reuse in Object-Oriented Web Engineering”, SAC2000, 2000

[HS94] Halasz, F., Schwartz, M.: “The Dexter Hypertext Reference Model”, Comm. of the ACM, vol. 37, pp. 30-39, 1994

[ISB95] Isakowitz, T., Stohr, E.A., Balasubramanian, P.: “RMM: A Methodology for Structured Hypermedia Design”, Comm. of the ACM, 1995

[JM98] Jörding, T., Meissner, K.: “Intelligent Multimedia Presentations in the Web: Fun without Annoyance”, WWW7, Brisbane, 1998

[Kl02] Klyne, G., Reynolds, F., Woodrow, C., Ohto, H.: “Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies”, W3C Working Draft, 2002.

[KW02] Koch, N., Wirsing, M.: “The Munich Reference Model for Adaptive Hypermedia Applications”, Second International Conference on Adaptive Hypermedia and Adaptive Web-based Systems. Springer Verlag, May 2002

[RSG01] Rossi, G., Schwabe, D., Guimaraes R.M.; “Designing Personalized Web Applications”, WWW10, Hong Kong (2001)

[SRB96] Schwabe, D., Rossi, G., Barbosa, S.D.J.: “Systematic Hypermedia Application Design with OOHDM”, UK Conference on Hypertext, 1996

[Sh96] Shen, W.M.: “An efficient Algorithm for Incremental Learning of Decision Lists”, Technical Report, USC-ISI-96-012, Information Sciences Institute, University of Southern California (1996)

[To94] Tochtermann, K.: “Ein Modell für Hypermedia”, PhD Thesis, Universität Dortmund, Fachbereich Informatik, Lehrstuhl 1 (1994)

[Wi01] Wireless Application Group: “User Agent Profile Specification“, WAP Forum (2001)

[WD02] Westbomke, J., Dittrich, G.: “Towards an XML-based Implementation of Structured Hypermedia Documents”, Journal of Universal Computer Science, Vol. 8/10 (2002)

[WS01] Wadge, W., Schraefel, M.: “A Complementary Approach for Adaptive and Adaptable Hypermedia: Intensional Hypertext” In Hypermedia: Openness, Structural Awareness, and Adaptivity - International Workshop OHS-7, SC-3, and AH-3, Aarhus, Denmark, 2001.