Design and Implementation of Component-based Adaptive Web Presentations

Zoltán Fiala Michael Hinz
Dresden University of Technology
Mommsenstr. 13, D-01062
Dresden, Germany
{zoltan.fiala, mh5}@inf.tu-dresden.de

Geert-Jan Houben Flavius Frasincar
Technische Universiteit Eindhoven
PO Box 513, NL-5600 MB
Eindhoven, The Netherlands
{g.j.houben, f.frasincar}@tue.nl

ABSTRACT

Engineering adaptive Web applications implies the development of content that can be automatically adjusted to varying client devices and user preferences. To meet this requirement, the AMACONT project recently introduced a component-based XML document format. Configurable document components encapsulating adaptive behavior and layout are used on different abstraction levels in order to support flexible reuse for effective Web page generation. This paper focuses on the process of designing and implementing such component-based adaptive Web presentations. Based on the model-driven specification framework from the Hera project, different phases of adaptation design are identified and their realization using AMACONT components is explained. Finally, a pipeline-based document generator for dynamically publishing component structures to different Web output formats is described.

Keywords

Component-based Web Engineering, Adaptive Hypermedia, Design Methods

1. INTRODUCTION

The WWW's change to a personalized ubiquitous medium of communication and cooperation necessitates the quick generation and delivery of up-to-date information that is automatically adapted to the appropriate presentation interface and user preferences. However, conventional document formats for the Web are hardly applicable for meeting this challenge. Their lack of support for the separation of content, structure, and layout prevents the flexible component-like reuse of fine-granular functional, semantic, and layout elements. Moreover, no mechanisms are provided for describing in a generic way the adaptive behavior and presentation of reusable pieces.

Recently, different approaches for modeling and engineering adaptive hypermedia and Web systems have emerged. As one of the most significant contributions we mention the AHAM reference model [1]. By capturing common abstractions of existing solutions, it provides a sound basis to describe, characterize, compare, and create adaptive hypermedia systems (AHS). A detailed survey on AHS development can be found in [3]. Different domains of AHSs, such as educational hypermedia, on-line information systems, information retrieval etc. are identified and representative systems are mentioned. Still, most solutions concentrate on the conceptual modeling and design process of adaptive hypermedia systems, not supporting the flexible reuse of adaptable implementation artefacts in a component-wise manner.

There are only a few approaches towards reusing implementation entities in hypermedia development. The WebComposition Markup Language [8] enables the component- based development of Web applications. Westbomke [15] proposes a formal XML-grammar for the implementation and presentation of platform-independent structured hypermedia documents. Still, adaptation to device capabilities and changing user preferences is not a central aspect of these approaches.

To address this problem, the AMACONT project recently introduced a component-based XML document format [5], [6]. It enables to compose adaptive Web applications by the aggregation and linkage of reusable document components that encapsulate adaptive content, behavior, and layout on different abstraction levels. Furthermore, a pipeline-based document generator for dynamically transforming adaptable component structures to different Web output formats was developed.

Although component-based reuse is crucial to Web Engineering, the development of adaptive Web and hypermedia applications out of components is a complex process that also necessitates to utilize systematic and disciplined design methodologies and specification frameworks. Such frameworks allow specifying hypermedia applications in an appropriate level of abstraction depending on both different stages of the engineering project (e.g. requirements analysis, design and implementation), and different dimensions of the problem area (e.g. data modeling, navigation modeling, presentation modeling) [14]. Recently, significant research on design and process models for hypermedia and Web applications has been done. Approved hypermedia design principles, such as those captured in OOHDM [13] or RMM [9],

SAC'04, March 14-17, 2004, Nicosia, Cyprus

have been even enhanced with the notions of adaptation and personalization in a further extension of OOHDM [12] or the RMM-based Hera methodology [7],[14].

Instead of suggesting yet another methodology, this work aims at the adoption of existing hypermedia design models for developing component-based adaptive Web applications. The main reason for this strategy is the observation that (due to the abstraction gap between high-level hypermedia design models and low-level (AMACONT) implementation entities) even different methodologies can be utilized to design component-based adaptive Web applications. The focus of this paper lies on using the model-driven Hera methodology [14] in the AMACONT context, allowing to develop data-driven adaptive Web applications (e.g. online-shops, e-galleries, Web sites of institutions etc.) from reusable components. Considering in a subsequent way the steps identified by Hera, it is shown how they can be applied to systematically realize adaptive Web presentations out of AMACONT implementation artefacts. One of the most interesting aspects is the question how different adaptation issues can be targeted in each design step.

The rest of this paper is structured as follows. Section 2 provides an overview of the component-based document format of the AMACONT project, especially focusing on how it supports adaptation. Along the lines identified by the model-driven Hera methodology, Section 3 deals with different phases of designing and implementing component-based adaptive Web applications. Section 4 introduces a pipeline-based modular document generator for dynamically transforming component structures to different Web output formats, and also explains how adaptivity can be supported during the document generation process. Finally, Section 5 concludes the paper and suggests future research directions.

2. ADAPTIVE WEB COMPONENTS

The component-based document format of the AMACONT project allows to build device-independent Web applications by aggregating and linking configurable document components. These components are documents or document fragments and instances of an XML grammar that represent adaptable content on different abstraction levels (Figure 1). The abstraction levels cover the full spectrum of granularities from single media items to coarse documents. Components' interfaces are described by metadata specifying their properties and their adaptive behavior. The format was defined using XML Schema.

2.1 Component Levels

On the lowest level there are media components encapsulating concrete media assets. These comprise text, structured text (e.g. HTML), images, sound, video, Java applets, and this media list may be extended arbitrarily. Besides technical properties described by MPEG-7 descriptors, additional content management information is provided, too. The second level combines media components belonging together semantically - e.g. an image with a textual description - into so called content unit components (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, thus abstracting from the exact resolution and presentation style of the current

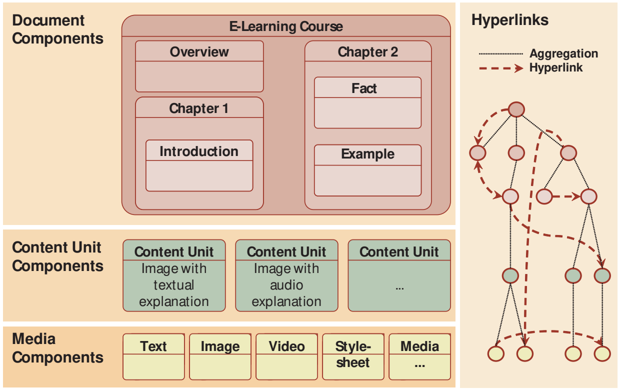

Figure 1: The document model.

display (see Section 2.2.2). 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 on the application context. Again, the spatial adjustment of subcomponents is described in a client-independent way. Finally, next to these three levels, 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.

2.2 Adaptation Support

The component-based document format aims at supporting adaptation by two mechanisms [6]. Firstly, it enables to encapsulate adaptation logic in components on different abstraction levels. Secondly, it allows describing the visual aspects of components by client-independent layout descriptors that can be automatically adapted to different output formats. In this section these two mechanisms are shortly summarized. Their utilization within the development process of adaptive Web presentations is explained in Section 3.

2.2.1 *Describing Adaptive Behavior*

To define adaptive behavior in a generic way, each component may include a number of variations. As an example, the definition of an image component might include (in its body) two variants 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 (see Section 4) 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 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-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. The processing XSLT stylesheet acts as an interpreter for this declarative "selection method language". Note that alternatives can be declared for components of all granularities, thus allowing to define adaptive behavior on different abstraction levels. A concrete example for defining component variants and selection methods will be shown in Section 3.2.

2.2.2 *Automatic Layout Adaptation*

As already mentioned, the document format allows 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), they 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 are defined. *BoxLayout* allows multiple components to be laid out either vertically or horizontally. *BorderLayout* arranges components to fit in five regions: *north*, *south*, *east*, *west*, and *center*. *OverlayLayout* allows to present components on top of each other. Finally, *GridLayout* enables to lay out components in a grid with a configurable number of columns and rows. Three XSLT stylesheets for automatically converting those descriptions to XHTML, cHTML and WML output were realized. A concrete example for defining adaptive layout will be shown in Section 3.3.

3. HERA-BASED DEVELOPMENT

The format introduced above provides Web engineers with reusable adaptive implementation artifacts. However, developing adaptive Web applications is a quite complex process that has to be based on systematic, disciplined methodologies and associated design models. By identifying crucial phases of Web development, an approach based on such design models helps designers and programmers to proceed in a structured way.

This section focuses on developing component-based adaptive Web applications according to the model-driven design methodology and specification framework of the Hera project. Hera is suitable for this undertaking for different reasons. Firstly, its main focus lies on the specification of highly adaptive Web presentations. Besides aspects of adaptability (adaptation during presentation generation to different device and user profiles), aspects of adaptivity (dynamic adaptation within the generated presentation itself) are also considered. Secondly, since it utilizes formalized XML descriptions, an automatic translation of Hera schemas to XML-based AMACONT components appears to be possible. According to the design phases identified by Hera, we can now show how those concepts can be utilized to systematically create, configure, aggregate, and link AMACONT components to complex adaptive Web applications.

3.1 Conceptual Design

The first step of the Hera methodology aims at representing the application domain using conventional conceptual modeling techniques. It results in the conceptual model (CM)

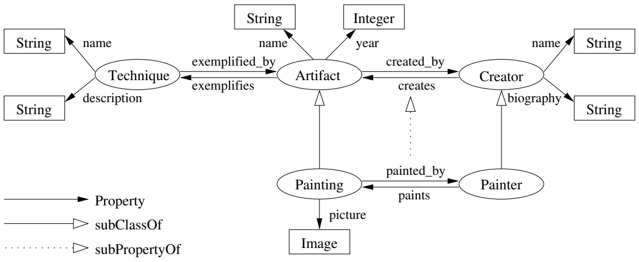

Figure 2: CM example.

consisting of a hierarchy of concepts, their attributes, and relationships. Concept attributes are typed. Besides basic types (e.g. $Integer$ and $String$), multimedia types (e.g. $Image$, $Audio$, $Video$) are also allowed, enabling to assign representative media items to concept attributes. The CM is expressed using RDFS [2]. Figure 2 depicts the conceptual model of a Web application presenting an online painting gallery.

When developing adaptive Web applications from AMACONT components, the conceptual design step as proposed by Hera has to be accompanied by the creation/retrieval of media instances representing concept attributes. Furthermore, in order to provide different device and output formats, component authors also have to reason about alternative media instances with different quality (concerning bandwidth, color depth, size etc.). According to Section 2, AMACONT prescribes that these media instances have to be encapsulated to media components (with variants) by describing them with corresponding MPEG7 descriptors.

3.2 Application Design

The application design step of Hera deals with the logical, structural, and navigational aspects of the Web presentation. The concepts introduced by the CM are translated into so called slices. A slice is a meaningful presentation unit of some media items, which can group both content attributes and other slices. There are two types of slice properties (relationships): slice composition (a slice including another slice) and slice navigation (a hyperlink abstraction between two slices). The most primitive slices represent concept attributes. The most complex ones (called top level slices) correspond to pages, which contain all the information present on the user's display at a particular moment. Figure 3 depicts a composite slice presenting the concept $painting$. The description of a $painting$ contains the actual $picture$, its $name$, the $year$ when it was painted, as well as (from the bottom slice) the information about its $painter$. Besides the graphical representation (also called Application Diagram) there is also an RDFS-based formalization of the AM.

3.2.1 *Adaptation design in the AM*

In order to adjust slice structures to device profiles and (changing) user preferences, Hera allows extending the AM with adaptation issues. Two types of adaptation or personalization are distinguished: *adaptability* and *adaptivity*. Adaptability (also known as static adaptation) means that the generation process is based on available information that describes the situation in which the user will use the generated presentation. Adaptivity (also mentioned as dynamic adaptation) is the kind of adaptation included in the generated adaptive hypermedia presentation. To put it simple, in the second case the hypermedia presentations themselves change while being browsed. This dynamic nature of adaptivity is supported by feedback mechanisms updating the user model according to the user's interactions with the presentation [1]. (Though not being a central issue of this paper, a mechanism for tracking user interactions in component-based adaptive Web applications will be explained in Section 4.1.)

Generally, different adaptation issues can be considered at application design. Firstly, it makes sense to adjust the coarse navigational structure to varying device (e.g. desktop computer, PDA, cell phone etc.) or user profiles. Depending on different device and user profiles one can decide which concepts should be presented and how they should be assigned to different interlinked slices. Secondly, the population of each specified slice with media items can be personalized, too. According to the media preferences and/or device capabilities of different users, different media types for presenting the same concept can be utilized. As an example, take the case of two customers, one of them preferring multimedia content, the other rather textual information. When presenting a painter's biography, the first one could be shown a video and an audio sequence, the second one a detailed textual description. Finally, dynamic adaptation (adaptivity) can also be targeted at this step. For example, different versions of a painter's biography could be presented in accordance with the user's changing knowledge on that painter: a long version at the user's first visit and a short one at his later visits.

In order to specify adaptation Hera prescribes that one associates appearance conditions to slice references [14]. These are Boolean conditions using attribute-value pairs from the user/platform. Two kinds of AM adaptation are enabled: conditional inclusion of slices and link hiding. Conditional inclusion means that a slice is included (and therefore visible) when it has a valid condition. Similarly, link hiding refers to the mechanism that a link is included when its destination slice is valid. Note that Figure 3 contains two appearance conditions: the first one (mentioning wap-phone) supports adaptability by forbidding the inclusion of *picture* for WAP phones: the use of *up* shows that the condition refers to the (static) user/device profile. The second one (mentioning biography) defines adaptivity by presenting different versions of a painter's biography depending on the user's knowledge on those painter (as captured in the user model, here refered to by *um*).

3.2.2 Realization with AMACONT components

There are important analogies between (the concepts of) Hera slices and AMACONT document components. Both represent meaningful presentation units bearing also some semantic role (e.g. painting, painting technique, newspaper article) and are recursive structures enabling an arbitrary depth of hierarchy. Moreover, both top-level slices and top-level document components correspond to pages to be presented on the users display. Finally, both may contain adaptation issues according to device profiles and user profile

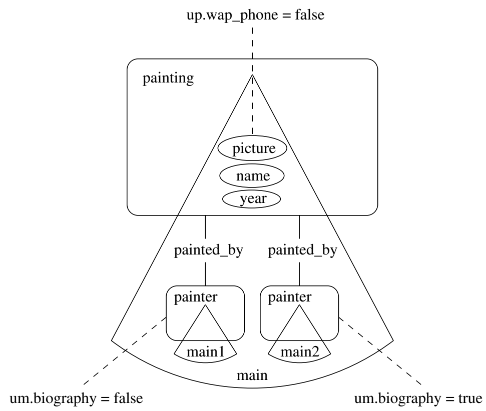

Figure 3: AM slice example.

parameters.

Nonetheless, there are also significant differences. Firstly, in contrast to slices, document components also encapsulate information on their layout. However, as application design does not deal with presentation issues, the specification of these layout properties can (and should) be postponed to a later stage of the development process (see Section 3.3). Secondly, whereas AM slices aim at defining the structure of presented concepts on the schema level (i.e. independent of concrete instances of those concepts), AMACONT components represent reusable implementation entities on the instance level. To bridge this gap it is meaningful to introduce the notion of AMACONT component templates. These are component skeletons that declare the structural, behavioral, and layout aspects of components independent of their concrete content. For example, a component author might create a generic component template for presenting painters. This template can be instantiated for specific paintings by being filled with media instances dynamically queried (retrieved) from a data source. We note that the use of a template mechanism is especially useful in data-driven Web Information Systems (WIS) and has already been utilized e.g. by the Web Modeling Language WebML [4].

The mentioned analogies allow component authors to specify the aggregation hierarchy of component templates according to the AM in a straightforward way. Firstly, top-level slices have to be mapped to top-level document components. Secondly, by unfolding slice aggregation relationships in a top-down manner, subslices and their attributes have to be mapped to subcomponents according to the following rules:

1. Concept attributes have to be mapped to media components. *Integer* and *String* attributes can be assigned to text components, media attributes to corresponding media components (image, audio, video etc.).
2. Slices containing concept attributes from only a single concept have to be mapped to single document components containing a content unit that aggregates the corresponding media components.
3. Slices referring to concept attributes and subslices from different concepts have to be mapped to composite

document components containing child document components for each aggregated subslice. For those subslices this mapping process has to be performed recursively.

As a final step, slice navigation properties have to be mapped to hyperlinks between component templates. The following XML code depicts the component aggregation hierarchy which was specified according to the slice example shown in Figure 3. Note that it still does not include any adaptation logic. (For better readability, there are some minor deviations between the syntax in this paper and the actual AMACONT XML-grammar. The *layer* attribute of components specifies whether they are document components (DC), content units (CU) or media components (MC).)

```
<PaintingComp name="Painting" layer="DC">
 ...
  <PaintingInfoComp name="PaintingInfo" layer="DC">
   ...
    <ImageAndTextComp name="PaintingAttr" layer="CU">
     ...
        <ImageComp name="PPicture" layer="MC"> ...
        <TextComp name="PName" layer="MC"> ...
        <TextComp name="PYear" layer="MC"> ...
    </ImageAndText>
  </PaintingInfoComp>
  <PainterComp name="Painter" layer="DC">
   ...
  </PainterComp>
</PaintingComp>
```

If the AM specifies adaptation aspects via appearance conditions, these have to be translated to AMACONT component variants and corresponding selection methods (as introduced in Section 2.2.1). This mapping can be also done in a straightforward way. Whenever a slice is provided with a Boolean appearance condition, the component assigned to its parent slice has to be split up into two variants. While the first variant includes the subcomponent assigned to the conditional slice, the second one excludes it. Furthermore, according to the slice condition a selection method in the IF-THEN-ELSE style has to be composed. As an example, following XML code depicts the component-based realization of the adaptivity example shown in Figure 3. Depending on (the state of) the user's knowledge different variants of a painter's biography are provided.

```
<PainterComp name="Painter" layer="DC">
  <MetaInformation>
    ...
  </MetaInformation>
  <Variants>
    <Variant name="Bio_Seen">
     ...
    </Variant>
    <Variant name="Bio_NotSeen">
     ...
    </Variant>
  </Variants>
</PainterComp>
```

The corresponding selection method can be specified in the header (*MetaInformation*) of that component as shown below:

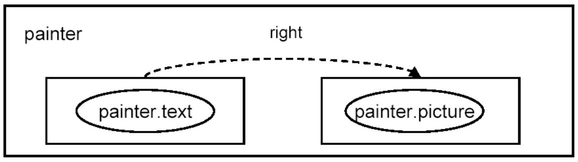

Figure 4: Hera presentation diagram example.

```
<MetaInformation>
 ...
  <AdaptiveProperties>
    <If>
      <Expr operator="equals">
        <UserParam>um.biography</UserParam>
        <Const>true</Const>
      </Expr>
      <Then res="Bio_Seen"/>
      <Else res="Bio_NotSeen"/>
    </If>
  <AdaptiveProperties>
 ...
</MetaInformation>
```

As a consequence, during document generation (see Section 4) the processing XSLT style sheet substitutes the variable *um.biography* by its Boolean value from the current user model, performs the selection method and determines the proper variant of the *PainterComp* component. As this variant might also have varying subcomponents, the style sheet works recursively. After the component variants to be included have been determined, link adaptation can be realized, too. As mentioned in Section 3.2.1, only links pointing to a valid (i.e. inserted) component or component variant are included.

3.3 Presentation Design

The presentation design step of Hera bridges the logical level and the actual implementation by introducing the implementation independent Presentation Model (PM). Complementary to the AM, where the designer is concerned with organizing the overall presentation structure and identifying what attributes from entities are to be included in slices, the PM specifies how and when the identified slices should be displayed. The PM is described in Hera by a presentation diagram (PD) consisting of regions and relationships between them [7]. During presentation design the slices introduced in the AM are mapped to regions. The presentation diagram specifies the organization of regions in a graphical way[1] by means of region relationships. Three types of region relationships exist: navigational, spatial, and temporal. As an example, Figure 4 depicts a possible presentation diagram for the *painter* slice by arranging the *text* and *picture* attributes in a horizontal way.

3.3.1 Realization with AMACONT components

The aggregation hierarchy of component templates was determined at application design, as we saw in Section 3.2.2. Now, (according to the guidelines of the PM) component authors are expected to specify the layout attributes of those component templates. As mentioned in Section 2.2.2, the

[1] At current time there is no XML-grammar for expressing PDs in a formal way.

component-based document format provides an XML-based mechanism for specifying the spatial adjustment of subcomponents within their container components in a size and client-independent way. Those abstract layout definitions support the automatic conversion of component structures to presentations in different output formats (such as WML, cHTML or xHTML). Thus, the spatial relationships between regions defined in the PM (or PD) have to be mapped to such component layout descriptions[2].

Again, this mapping happens in a straightforward way. Beginning at top-level document components and visiting their subcomponents recursively, one has to declare for each component (variant) how its immediate subcomponents are arranged. The appropriate layout descriptors are added to the meta-information section of each component's header. As an example, the presentation diagram shown in Figure 4 can be mapped to the layout manager of the *PainterInfo* component template according to a horizontal *BoxLayout* scheme:

```
<MetaInformation>
 ...
  <LayoutProperties>
    <BoxLayout orientation="xAxis">
      <ComponentRef ratio="30%" name="PainterText">
      <ComponentRef ratio="70%" name="PainterImage">
    </BoxLayout>
  </LayoutProperties>
 ...
</MetaInformation>
```

Note that there are also other adaptation aspects to be considered during presentation specification. Firstly, since today's client devices are characterized by varying technical parameters and support different output formats, the inserted media components have to be also adjusted based on their qualitative properties, such as bit rate, color depth, resolution etc. Secondly, depending on the layout preferences of different user stereotypes (children, adults, visually impaired users etc.), different layout elements determining the corporate design of the Web presentation (e.g. font sizes, logos, buttons, background images etc.) have to be provided, too. In order to consider such adaptation aspects, additional variants for selected media components as well as corresponding selection methods have to be specified. Note that such variants can be defined even after the declaration of layout managers: since layout managers of a given component reference only its immediate subcomponents, these subcomponents may have variants according to different adaptation aspects, too.

4. DOCUMENT GENERATION

The previous section dealt with the engineering process of adaptive component-based Web presentations. According to the Hera methodology, different phases of adaptation design were identified and their realization with AMACONT components was explained. This section focuses on the process of document generation and describes how the developed

[2]As the AMACONT document format does not support expressing temporal presentation aspects, we currently limit presentation design to only navigational and spatial aspects.

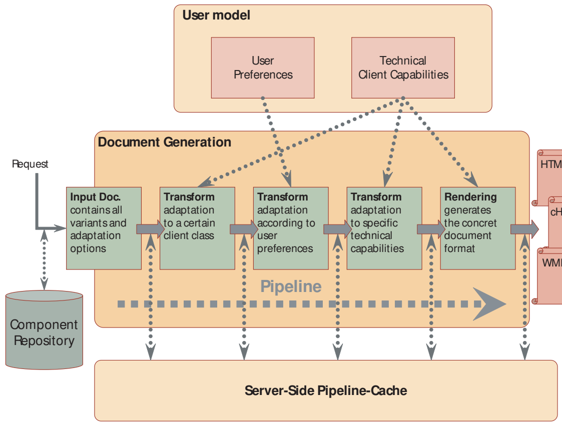

Figure 5: Pipeline-based document generation.

AMACONT components are dynamically adjusted to varying user preferences and client devices.

Document generation is based on a stepwise pipeline concept, as illustrated by Figure 5. The input of the document generator is a complex component encapsulating all possibilities concerning its content, layout, and structure. It is created by instantiating a component template according to a dynamic user query. According to the user/platform profile (which is stored on the server according to a CC/PP [11] vocabulary), it is subdued to a series of XSLT transformations, each considering a certain adaptation aspect by means of the configuration and selection of component variants.

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 5 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). Although 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. E.g. when the same document is requested by two PDA users, the output of this transformer 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 (e.g. XHTML, cHTML, WML) is generated. This concrete rendering of components happens automatically. The document generator was realized based on the publishing framework Cocoon [16].

4.1 Supporting Adaptivity During Document Generation

The document generation architecture as shown in Figure 5 realizes adaptability. It adjusts input document components according to different device profiles and user model parameters. However, in order to provide adaptivity, too, this architecture has to be extended. Firstly, a mechanism for triggering user model updates according to user interactions is required. Secondly, an adaptation engine performing those user model updates has to be utilized. In order to track user interactions (following links, starting videos, interrupting downloads etc.), client-side mechanisms based on JavaScript routines were developed. These can then be assigned to components at authoring time and are automatically inserted and configured during document generation. They gather information on user interactions and send it to the server according to a specific CC/PP profile [6]. By evaluating interaction history lists, suggestions on users' preferences and knowledge can be made and parts of the user model can be updated or specialized. In a developed prototype application for product presentation this specialization is performed by the incremental learning algorithm CDL4. In the TELLIM project [10] we proved that this algorithm was extremely useful in adaptive multimedia product presentations. However, since the CC/PP profiles for storing interactions, user models, and device capabilities provide a well-defined interface, different adaptation engines may be utilized in the future.

5. CONCLUSION AND FUTURE WORK

In this paper we have demonstrated how adaptive Web presentations can be constructed in a component-based way, combining the strengths of the model-driven design methodology and specification framework of the Hera project and the component-based XML document formats of the AMACONT project. We have shown how during the phases of design and implementation, different aspects of adaptation can be dealt with. We have illustrated how both adaptability and adaptivity can be considered in the design and how subsequently a pipeline-based document generator can implement the process of dynamically publishing component structures in different output formats. As matters of future work, we mention:

- the semi-automatic translation of Hera's XML schemas to AMACONT component templates
- extending the Hera framework with the AMACONT pipeline in order to utilize its presentation generation capabilities
- the construction of a graphical tool that supports the creation of Presentation Diagrams and outputs AMACONT components
- the specification of user interaction within the design models

6. REFERENCES

[1] P. D. Bra, G. J. Houben, and H. Wu. AHAM: A dexter-based reference model for adaptive hypermedia. In *HYPERTEXT '99, Proceedings of the 10th ACM Conference on Hypertext and Hypermedia, Darmstadt, Germany*, pages 147–156. ACM, 1999.

[2] D. Brickley and R. Guha. RDF *Vocabulary Description Language 1.0:* RDF *Schema.* W3C Working Draft, 10 October 2003.

[3] P. Brusilovsky. Adaptive hypermedia. *User Modeling and User Adapted Interaction*, 11:87–110, 2001.

[4] S. Ceri, P. Fraternali, and A. Bongio. Web modeling language (webml): a modeling language for designing web sites. In *9th International Conference on the World Wide Web (WWW9), Amsterdam*, 2000.

[5] Z. Fiala, M. Hinz, K. Meissner, and F. Wehner. A component-based approach for adaptive dynamic web documents. *Journal of Web Engineering, Rinton Press*, 2(1&2):058–073, September 2003.

[6] Z. Fiala, M. Hinz, and F. Wehner. An XML-based component architecture for personalized adaptive web applications. In *Workshop Personalisierung mittels XML-Technologien, Berliner XML Tage*, pages 370–378, 2003.

[7] F. Frasincar, G. J. Houben, and R. Vdovjak. An RMM-based methodology for hypermedia presentation design. In *Advances in Databases and Information Systems, 5th East European Conference, ADBIS 2001, Vilnius, Lithuania*, pages 323–337, 2001.

[8] M. Gaedke, C. Segor, and H.-W. Gellersen. WCML: Paving the way for reuse in object-oriented web engineering. In ACM *Symposium on Applied Computing (SAC2000)*, 2000.

[9] T. Isakowitz, E. Stohr, and P. Balasubramanian. RMM: A methodology for structured hypermedia design. *Comm. of the ACM*, 38(8):34–44, 1995.

[10] T. Jörding and K. Meissner. Intelligent multimedia presentations in the web: Fun without annoyance. In *Seventh International Conference on the World Wide Web (WWW7), Brisbane, Australia*, 1998.

[11] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M. Butler, and L. Tran. *Composite Capability/Preference Profiles (*CC/PP*): Structure and Vocabularies.* W3C Working Draft, 2003.

[12] G. Rossi, D. Schwabe, and R. Guimaraes. Designing personalized web applications. In *WWW10, The Tenth International Conference on the World Wide Web, Hong Kong*, 2001.

[13] D. Schwabe, G. Rossi, and S. D. J. Barbosa. Systematic hypermedia application design with OOHDM. In *Hypertext '96, The Seventh ACM Conference on Hypertext, Washington DC, 1996*, pages 116–128. ACM, 1996.

[14] R. Vdovjak, F. Frasincar, G. J. Houben, and P. Barna. Engineering semantic web information systems in hera. *Journal of Web Engineering, Rinton Press*, 2(1&2):003–026, 2003.

[15] J. Westbomke and G. Dittrich. Towards an xml-based implementation of structured hypermedia documents. *Journal of Universal Computer Science*, 8(10):944–956, 2002.

[16] C. Ziegeler and M. Langham. *Cocoon: Building XML Applications.* New Riders, 2002.