Journal of Web Engineering, Vol. 2, No.1&2 (2003) 058-073
© Rinton Press

A COMPONENT-BASED APPROACH FOR ADAPTIVE DYNAMIC WEB DOCUMENTS

ZOLTÁN FIALA, MICHAEL HINZ, KLAUS MEISSNER, and FRANK WEHNER

Dresden University of Technology, Heinz-Nixdorf Endowed Chair for Multimedia Technology
[zoltan.fiala, michael.hinz, klaus.meissner, frank.wehner]@inf.tu-dresden.de

Received April 8, 2003

Personalized Web applications automatically adapted for different clients and user preferences gain more importance. Still, there are barely technologies to compensate the additional effort of creating, maintaining and publishing such Web content. To address this problem, this paper introduces a declarative, component-based approach for adaptive, dynamic Web documents on the basis of XML-technology. Adaptive Web components on different abstraction levels are defined in order to support effective Web page authoring and generation. Media components encapsulate concrete media assets by describing them with technical metadata. Content units group media components belonging together semantically by declaring their layout in a device-independent way. Finally, hierarchical document components playing a specific semantic role are defined. The hyperlink view for defining typed links is spanned over all component layers. Beside the reuse of both implementation artefacts and higher level concepts, the model also allows to define adaptive behavior of components in a fine-granular way. As a further benefit the support for ubiquitous collaboration via component annotations is introduced. Finally, the stepwise pipeline-based process of document generation is introduced and performance issues are sketched.

Key words: Web Engineering, Component-based Development, Adaptive Hypermedia, Document Generation, Web Annotations, Caching
Communicated by: M Gaedke & C Watters

1 Introduction

Recently, the amount of information published on the World Wide Web has grown enormously. The survey in [30] reports of 21 Terabytes of Web content published in 2001 and prognosticates an annual doubling for the future. Another important trend is the change of the Web from a presentation to a communication and cooperation medium. While formerly it was a collection of static HTML-pages, interactive and personalized Web applications availing the bidirectional nature of the WWW become very important today. A further tendency is the universal access to the Web via different multimedia devices. Beside desktop PCs the usage of mobile devices, such as cell phones, PDAs, and notebooks is rapidly increasing.

These trends necessitate the quick generation and delivery of up-to-date information, which is automatically adapted to the respective presentation interface and user preferences. Still, most content providers today use to create and manage content for different clients separately. In order to compensate for this additional effort, changes regarding the conception, creation, updating, and publishing of Web presentations within a comprehensive Web content management are needed.

Conventional document formats for the Web, such as HTML, WML, cHTML, and XHTML have significant disadvantages concerning their applicability for managing adaptive, dynamic Web presentations. The main reason for this is the lacking separation of structure, content, and layout [24], not allowing to uniformly create and update content for different platforms at once. A further problem is the coarse-grained implementation model of the Web [18]. Fine-granular functional, semantic and layout elements get lost during the implementation phase while being transformed into file-based resources such as HTML, cHTML, WML documents etc. Reuse and effective caching of independent, adaptive document fragments is not possible. Furthermore, existing formats describe the semantic meaning of Web pages or their parts only insufficiently and do not allow their maintenance independent of a certain output format.

To overcome this problem, this paper introduces a component-based document model for efficiently designing, creating, maintaining, and delivering adaptive, dynamic Web presentations. In section 2 existing research approaches are analyzed and compared. Then the concept of Web document components is defined and the document model is introduced in detail. Section 4 sketches selected benefits of the model, such as component reuse, support for adaptive page generation and collaboration using fine-granular annotations. Finally, Section 5 deals with the details of the pipeline-based concept of page generation and also concerns selected performance issues.

2 Related Work

2.1 Modeling Adaptive Web Documents

Brusilovsky [5] defines adaptive hypermedia systems (AHS) as "*all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user*". In [6] he considers adaptable parts of content, layout and navigation to be adjusted both to user characteristics (such as knowledge, background, learning style etc.) and the environment (e.g. the user's location and platform).

There exist a number of approaches to model adaptive hypermedia documents. In [38] an approach called „Intensional Hypertext" is introduced. The „Intensional Markup Language" (IML) [35] is a proprietary extension of HTML containing control structures for adaptive behavior on the basis of the chosen URL and user input. However, there is no support for output formats other than HTML. An adaptive document generation system based on a matrix of contents is introduced in [28]. Documents are modeled as hyper graphs defined by nodes (concepts, procedures, principles, and facts) and relations between them (instantiation, composition, specialization, and precedence). A matrix is used to map domain models and user models. [22] presents an adaptive travel application on the basis of XML technologies like XForms, XHTML and DOM using the MVC paradigm. Still, adaptation in this system concerns only the client platform but no user preferences. The TELLIM-project [25] presents a prototype for adapting content and layout of product presentations according to temporal user models. Unfortunately, there is no support to author general-type adaptive Web applications for different platforms.

In order to describe, characterize and compare AHSs in a common formal way, De Bra et al. introduced the AHAM reference model [4], which bases on the previous Dexter model [21] and distinguishes between the *domain*, the *user* and the *teaching* model of an AHS. The *domain model* describes how the information is structured and linked together by defining *atomic concept components* (primitive fragments of information being not adaptable), *composite concept components* (containing a number of atomic or composite concept subcomponents) and *concept relation components*. Adaptation occurs by including or excluding page fragment components as well as hiding or annotating links according to rules of the *pedagogical model* parameterized by the *user model*. However, a component in AHAM is rather an abstract representation form of information and not a reusable, configurable implementation unit with well-defined attributes and a concrete layout specification. Moreover, a page concept may contain only atomic concepts, i.e. only a flat hierarchy of fragments for page composition is allowed. As an implementation of the reference model, the AHA! system is introduced in [10], [11]. However, mobile end devices, Web output formats different from HTML, as well as configuration and rearrangement of reusable page fragments in a fine-granular way have not been concerned yet.

In general, there is still no sufficient support for efficiently authoring general-type, ubiquitous adaptive Web applications. The current coarse-grained, resource-based implementation model of the Web does not allow the reuse of both adaptable implementation artefacts (document fragments, media assets etc.) and higher level concepts (such as adaptation rules or abstract layout information) in a flexible, component-based way. To solve this problem, this paper aims at introducing a component-based approach for adaptive Web applications. Beforehand, current research on object- and component-based Web engineering is summarized.

2.2 Object- and Component-oriented Web Engineering

Recently, a lot of research on object- or component-based Web engineering has been done. The Object-Oriented-Hypermedia-Design-Model (OOHDM) [36] is a process model for the design of complex hypermedia applications defining the four phases *conceptual design*, *navigational design*, *abstract interface design*, and *implementation*. Aim of the model is the reuse of existing design concepts. However, it does not consider their mapping to reusable implementation artefacts. A similar design principle is proposed by the Relationship Management Methodology (RMM) [23] focused on creating data-intensive Web applications. Unfortunately, reuse of implementation entities in other processes is not supported, either.

A common drawback of these methodologies is their lacking support for personalization and adaptation. Therefore, approaches for designing personalized Web applications with OOHDM [34] as well as an RMM-based specification framework for engineering adaptive Web applications [15] have been introduced. Still, these works concentrate on the specification of adaptive Web presentations, too, and do not focus on reusing configurable, adaptive implementation artefacts.

The modeling language WebML [7] introduces an XML-based notation for the specification of complex data-driven Web sites at the conceptual level. Its *structure*, *hypertext* and *presentation model*s strongly resemble both the design phases of OOHDM and the steps of RMM. Furthermore, an additional *personalization model* for defining user- and group specific Web presentations via event-condition-action rules is introduced. Mapping of design concepts to Web pages is done by XSLT-stylesheets. However, reuse of implementation artefacts and adaptation at the presentation level – e.g. to varying technical capabilities of client devices - are not considered.

In [17], [18] Gaedke et al. propose the application of component-oriented methods from software engineering to the development of high-quality Web applications. For this reason they introduce the WebComposition Process Model and the WebComposition Markup Language (WCML) to define components modeling Web entities with respect to a variety of target languages and of arbitrary granularity. Reuse of design artefacts and automatic mapping of information items to the implementation model via so called decorators is supported. In [16] the WebComponent Repository as a key tool for systematic component reuse is introduced. Though aspects of personalization are described in detail in [20], adaptation is not a central aspect of the approach.

The project CHAMELEON [39] aims at developing formats and tools for reusable, adaptive courseware (courseware components) for Web-based learning systems. Courseware is represented in an XML-based document format called TeachML distinguishing between four abstraction levels of courseware. On the lower level external media objects described by media specific properties are referenced. The next level combines them to semantic content units, e.g. a media object with according explanation. On the third level didactical units representing application context, such as chapter, proof etc. are introduced. Finally, on the upper level there are didactical structures to define navigational paths, hyperlink views etc. Adaptation concerns both the adjustment of navigational paths to the learners' knowledge and the presentation of components to different output formats.

The document model proposed in this paper was inspired by the declarative, document-centric, level-based component approach introduced by CHAMELEON and tries to generalize it for a broader range of adaptive Web applications. In contrast to the mentioned process models this paper focuses not on the conceptual design of Web applications, but on the challenge to reuse adaptable implementation artefacts in Web presentations.

3 The Document Model

In this section a component-based document model for adaptive Web sites is introduced. The model intends to support effective Web page authoring and generation. Firstly, the general concept of Web document components is explained, and then the level-based architecture of the model is described.

3.1 Declarative Web Document Components

In this model Web sites are composed of configurable Web components. These are documents or fragments of documents (instances of a specific XML-grammar) that represent adaptable Web content on various abstraction levels. Components are described by attached metadata acting as the interface definition specifying their properties and adaptive behavior. For example, an image object provided with information on its technical characteristics and different presentation styles for various clients is a component in this model. Web sites are constructed by aggregating and linking components to complex document structures. During Web page generation - which is described in detail in Section 5 - these abstract document structures are translated into Web pages in a concrete output format adapted to a specific user model or client, respectively.

Even though document fragments are no (binary) software components according to the software engineering definition [37], they show a lot of similarities to the classic component concept. Firstly, they are system independent, reusable, and extensible units, representing a certain functionality that can be combined to complex applications. Furthermore, they provide a clearly defined interface described by specific metadata for configuration and aggregation on higher component levels.

The implementation of the document model is based on XML-technology in order to support platform independence and interoperability. In contrast to [18] and [7] the document format is defined by XML Schema which allows for more powerful definitions than a DTD and provides for namespaces, reuse of type definitions, type safety and more exact expressions of cardinality.

3.2 The Level-based Architecture

In Figure 1 the level-based architecture of the document model is shown.

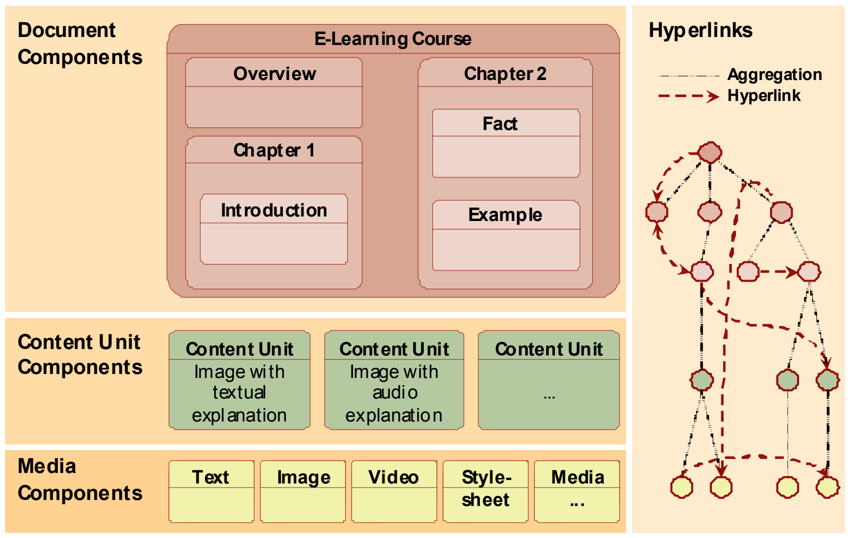

Figure 1: The Document Model

On the lowest level there are *media components* encapsulating particular media assets described by specific technical and non-technical metadata. The set of media assets comprises text, structured text (e.g. HTML or XML code), images, sound, video, java applets, Flash presentations but may be extended arbitrarily. Dynamically created media assets (e.g. HTML pages or pictures generated on the fly) are also supported, provided that corresponding metadata is delivered, too. It has to be emphasized that even whole documents (such as HTML-documents) might be wrapped to media assets in order to optimize page generation and support the authoring process. Technical metadata (e.g. size, format or resolution of an image) is represented by MPEG7-descriptors [31]. Non-technical metadata describing high-level characteristics - such as adaptivity, caching-behavior or component annotations etc. - is independent from media type and expressed according to the XML-grammar. Finally, descriptive content management information - ownership, access rights etc. - is also an important part of the metadata scheme. The following example depicts how technical metadata based on MPEG7-descriptors is represented.

```
<MediaComponent name="Video">
  <MetaInformation>
    ...
    <TechnicalProperties>
      <mpeg7:Video>
      ...
        <mpeg7:Format>
          <mpeg7:Name>MPEG-1 Video</mpeg7:Name>
        </mpeg7:Format>
        <mpeg7:Pixel aspectRatio="0.75" bitsPer="8"/>
        <mpeg7:Frame height="288" width="352" rate="25"/>
        ...
      </mpeg7:Video>
    </TechnicalProperties>
  </MetaInformation>
</MediaComponent>
```

On the next level media components belonging together semantically are combined to so called *content units*. For example, an image with according textual description can constitute a content unit. The definition of such collections of media objects is a key factor of component reuse. Predefined content units are e.g. *media object with explanation*, *list of media objects*, *table containing media objects*, but arbitrary extensions are supported.

Beside non-technical metadata introduced for media components, additional layout properties can be defined in order to describe the spatial adjustment of contained media objects within a content unit. Inspired by the layout manager mechanism of the Java language, these properties describe a size- and client-independent layout (*BoxLayout*, *OverLayLayout*, BorderLayout, *GridTableLayout* etc.) 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. Following code snippet depicts the layout description of a content unit containing two media objects adjusted by the layout manager *BoxLayout*. An image and a text object are arranged above each other, taking 30 and 70 percent of the available vertical space.

```
<MetaInformation>
    <LayoutProperties>
          <LayoutManager>
               <BoxLayout orientation="yAxis">
                     <ComponentRef ratio="30%">
                            PictureObject1
                     </ComponentRef>
                     <ComponentRef ratio="70%">
                            TextObject1
                     </ComponentRef>
               </BoxLayout>
          </LayoutManager>
    </LayoutProperties>
</MetaInformation>
```

The next abstraction level specifies *document components*. These are parts of Web presentations playing a well defined semantic role, such as a news column, a product presentation or even a whole Web site. Take for example a content unit containing an image object and a text object. It could be encapsulated to a document component and be given the semantic role "newspaper article". Document components can not only reference content units, they can also aggregate other document components. This aggregation results in a hierarchy of document components which describes the logical structure of a Web site and is strongly dependent from the application context. In analogy to content units, the presentation of document components on a Web page is specified by layout properties defining the spatial adjustment of other aggregated document components. However, these properties only describe the adjustment of subcomponents within the container component. The information about the layout of subcomponents is described in themselves. Thus, in order to provide reuse and configuration, each component stores and manages its layout information on its own. The following example demonstrates the aggregation of components.

```
<AmaCourseComponent name="E-LearningCourse" layer="DocumentComponent">
  <SubComponents>
    <AmaOverviewComponent name="Overview" layer="DocumentComponent">
      ...
    </AmaOverviewComponent>
    <AmaChapterComponent name="Chapter1" layer="DocumentComponent">
```

```
      <SubComponents>
        <AmaIntroductionComponent name="Introduction" layer="DocumentComponent">
          <SubComponents>
            <AmaImageTextComponent name="ImageWithDescription" layer="ContentUnit">
              <SubComponents>
                <ImageComponent name="Image" layer="Media">
                  ...
                </ImageComponent>
                <TextComponent name="Text" layer="Media">
                  ...
                </TextComponent>
              </SubComponents>
            </AmaImageTextComponent>
          </SubComponents>
        </AmaIntroductionComponent >
      </SubComponents>
    </AmaChapterComponent>
    ...
  </SubComponents>
</AmaCourseComponent>
```

Whereas aggregation relationships between components are expressed on the level of document components, links between fine-granular components on arbitrary levels are defined on the *hyperlink view*, which is spanned over all component levels. That is to say, hyperlinks are stored separately from the referenced content. Uni- and bidirectional typed links based on the standards XPath and XPointer are allowed. The source anchor of a link is unequivocally described by an XPath-expression identifying the source component and a corresponding XPointer expression defining the offset of the link in that component. Link destinations can be arbitrary components identified by XPath, too. The following code example demonstrates a unidirectional link pointing from one component to another.

```
<Hyperlinks>
    <AmaComponentLinkComponent name="l1" layer="Hyperlink">
        <From pointer="xpointer(string-range(documentComp_18,'',14,8))"/>
        <To component="documentComp_19"/>
    </AmaComponentLinkComponent>
</Hyperlinks>
```

4 Model Benefits

In this section, major advantages and possible usage scenarios of the proposed document model, namely component reuse, support for adaptive page generation and collaboration via fine-granular Web annotations are shown. While elaborating these benefits, comparisons to existing research issues are drawn.

4.1 Component Reuse

A very important aspect of effectively engineering Web sites is the reuse of formerly developed artefacts. Still, the coarse-grained document-oriented implementation model of the Web makes it difficult for authors to identify and reuse configurable content fragments of a Web presentation [18]. The document model presented in this paper tries to solve this problem by defining fine-granular Web artefacts for composing Web presentations. By setting appropriate properties existing Web components may be reconfigured for reuse in different application contexts.

The level-based structure of the document model supports the effective reuse of components of a certain level on higher levels. For example, a content unit arranging a list of pictures in a table might be reused by being filled with different image components. Furthermore, the recursive nature of document components allows for reuse of component trees of arbitrary depth.

The vision is a Web-based component store, where authors of Web presentations may create, store, configure, and retrieve generic Web components and corresponding transformers for Web page generation.

4.2 Support for Adaptive, Personalized Web Applications

The approach based on the proposed document model addresses different aspects of adaptation. Firstly, adaptation of content to different clients using different document formats is supported. Secondly, various technical preferences, such as average network bandwidth, display resolution, installed Web browser plug-ins etc. are considered. Finally, it is intended to express user models and adaptation rules for semantic, application-specific personalization. It is important to emphasize that adaptive behavior is not only described for concrete content pieces of a Web presentation but also on higher abstraction levels. The idea is to attach metadata describing adaptive behavior to components on different abstraction level.

4.2.1 Adaptation on Different Abstraction Levels

Firstly, adaptation is required on the lower level of media objects in order to consider various client capabilities or other technical preferences. For instance, in a personalized Web presentation it is necessary to provide different instances of a certain picture with variable size, color depth or resolution in order to automatically adapt to various display types.

Adaptation is necessary on the level of content units, too. On one hand, technical properties of client devices can be taken into account. For example, in an online-shop application a user with a high performance client could be shown a short video clip of the presented article, while others with low-performance displays would be presented an image of the product. On the other hand, the adaptation of content units may also rely on semantic user preferences. Consider again the case of two customers, one of them preferring detailed textual descriptions, the other visual information. In this case, the presentation for the first user includes content units primarily referring to text objects, and for the other one content units referring to images, respectively. These kinds of adaptation were approved as profitable in the TELLIM project [25].

Adaptation of document components concerns the component hierarchy, i.e., the way components are nested. This results in different variations of component trees depending on user preferences and client properties. This mechanism is shown in Figure 2 by an example taken from a learning application. The generated courseware contains different didactical components (e.g. introduction, facts, examples), depending on varying learning styles and previous knowledge of students. In this special case document components requiring basic knowledge are not included initially. However, when basic knowledge is already present, the introduction can be omitted and more detailed information can be presented. Furthermore, the adjustment of document components may also depend on client capabilities. Let us consider the Web portal of a railway company as an example. Whereas the desktop PC version of such a Web site might include numerous columns such as timetable, online-shop, online travel agency etc., the WAP version is mostly restricted to a minimal set of services, e.g. merely an online timetable.

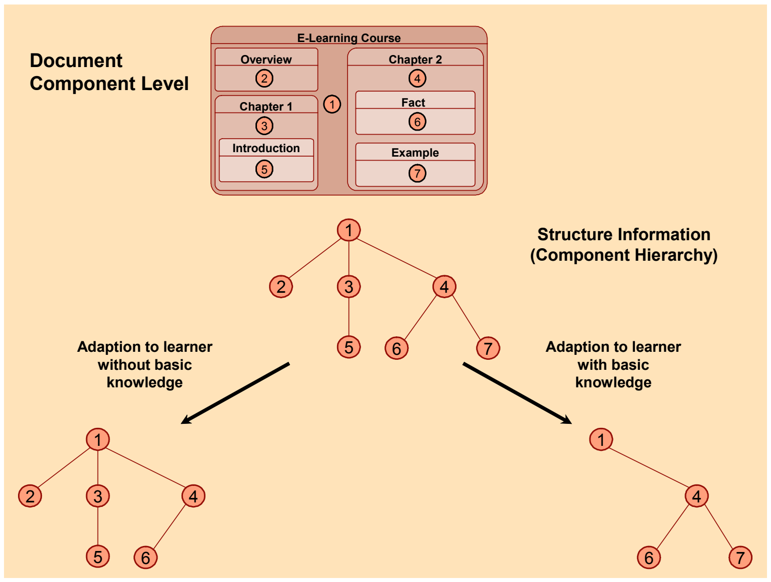
Document Component Level
E-Learning Course
Overview
Chapter 1
Introduction
Chapter 2
Fact
Example
Structure Information (Component Hierarchy)
Adaption to learner without basic knowledge
Adaption to learner with basic knowledge

Figure 2 Adaptation of Web Components

Hyperlinks create relations between components and define a network over the component structure that can be adapted to users or user groups. Thus, in order to allow link adaptation, properties concerning adaptation can be set on the hyperlink view, too. This results in variable hyperlink structures for different users allowing adjusted navigation through the Web presentation.

4.2.2 Describing Adaptive Behavior

In order to achieve adaptation of components they are provided with specific metadata describing their adaptive behavior. Concretely, each component definition may include a number of variants. As an example, the definition of an image object component might include two variations, one for color and another for monochrome displays. Similarly, the number, structure and arrangement of subcomponents within a document component could also have different variants according to a certain adaptation aspect. The following example shows how variants for an image component with varying color-depths can be expressed.

```
<AmaImageComponent name="Product_Image" layer="Media">
    <MetaInformation>
        ...
    </MetaInformation>
    <Variants>
        <Variant name="MonoChrome" layer="Media">
            ...
        </Variant>
        <Variant name="16color" layer="Media">
            ...
        </Variant>
        <Variant name="TrueColor" layer="Media">
            ...
        </Variant>
    </Variants>
</AmaImageComponent>
```

The decision, which alternative is chosen and inserted into the current Web presentation is always made "on-the-fly" during document generation by a corresponding transformer stylesheet implementing a certain selection method. The author of a Web application selects the appropriate adaptation method from a repertoire of such methods. Each of them is parameterized by the current user model, which contains information on client capabilities as well as on the user's semantic preferences. This separation of describing component variants (in the component itself) and adaptation logic (in stylesheets) allows the reuse of a given component in different adaptation scenarios.

The internal representation of selection methods relies on a specific XML-grammar, too. This enables the declaration of constants, variables representing user model parameters, terms based on constants, variables and operators, as well as complex conditional expressions. The following example depicts a possible selection method for the image component shown above according to the current display's color depth. Note that attributes from the user model are accessible as variables (UserParam). The "C-style" pseudo-code on the right aims the better understanding of the XML code on the left.

```
…
<MetaInformation>
   …
   <AdaptiveProperties>
      <Switch>
         <Expr>
            <UserParam>bitsPerPixel</UserParam>
         </Expr>
         <Cases>
            <Case value="1" res="MonoChrome"/>
            <Case value="4" res="16Color"/>
            <Case value="24" res="TrueColor"/>
            <Default res="16Color"/>
         </Cases>
      </Switch>
   <AdaptiveProperties>
   …
</MetaInformation>
…
```

```
switch (userParam.bitsPerPixel)
{
   case 1: choose(MonoChrome); break;
   case 4: choose(16Color); break;
   case 24:choose(TrueColor); break;
   default: choose(16Color);
}
```

During document generation the variable bitsPerPixel is substituted by the current user's display's color depth and the corresponding variant is determined.

In order to describe client devices in a standardized way CC/PP (Composite Capabilities/Preference Profiles) [26] is used. CC/PP is based on RDF [29] and allows to express device capabilities and user preferences that can be used to guide the adaptation of content presented to that device. The default CC/PP attributes of UAProf [3] for describing device capabilities were extended by additional descriptors in order to represent information gathered on client-side user interactions, such as following a certain link, starting audio or video players, interrupting downloads etc. Moreover, semantic user preferences are also stored based on RDF on the server-side.

In Section 5 the process of document generation will be explained in more detail.

4.3 Collaboration Support by Reusable Fine-Granular Annotations

Annotating Web pages is an important aspect of asynchronous communication on the WWW. Authors and visitors of Web applications attach notes to certain pieces of Web content in order to remember things better, to communicate with each other or to manage information more intelligently. Typical scenarios of Web annotations are Web-based learning systems allowing students and tutors to communicate, distributed authoring environments supporting the concurrent editing of content, and even product presentations, where users give feedback to the system via personal remarks.

Existing annotation environments, like ComMentor [33], Annotator [32], CritLink [41], CoNote, YAWAS [12], Virtual Notes [19], iMarkup [1] etc. mainly focus on annotating static Web pages which do not change their content, structure and layout temporally. In general, an annotation is clearly defined by the URL of the Web page containing it and some anchor points within that page [13].

A significant disadvantage of these tools is the lacking support for separation of content, structure, and layout. Annotations are not attached to the contents itself, rather to the Web pages containing them. Notes go lost, when the same content is presented on a different page, in a new context or with a changed layout. Thus, this mechanism is not suitable for dynamic Web documents generated at run-time for which no persistent state and no constant layout exists.

The proposed document model supports the creation of annotations to reusable components. That is to say, the smallest objects to be annotated are not whole Web pages but document fragments, i.e. *media components*, *content units* or *document components*. When a user marks a Web page generated on the basis of Web components, his annotations are *reversely mapped* to the fine-granular content components and stored as specific component metadata. When the same component is used in another presentation - possibly on a different client or in a different context - the attached annotations can be reused, too. Furthermore, annotations themselves are not only restricted to textual notes or images, they can be arbitrary components like pieces of text, images or even whole Web pages which can be annotated again. Consequently, annotations can be seen as typed hyperlinks allowing personalized navigation in the component space. Annotation anchors are unequivocally located by the identifier of the corresponding component and some offset coordinates within it. For annotations of text fragments such offset coordinates are expressed by the string range mechanism of the XPointer-language allowing layout independence. The concept of attaching fine-granular annotations to reusable, adaptive components was successfully implemented in a prototype called *DynamicMarks* [14].

Annotating Web components on different abstraction levels addresses various usage scenarios. Firstly, attaching textual remarks or individual references to arbitrary Web components - such as linking a picture object from a product presentation to a certain column of an electronic newspaper - allows for a personalized media asset and knowledge management for any individual user. Secondly, exchanging annotations to fine-granular content fragments is a very important way of collaboration. Let us take the case of a Web-based learning system, where a certain mathematical course for students is presented. A document component representing the Theorem of Pythagoras might be presented in a different context and layout for two students according to their varying mathematical skills and/or the client systems technical parameters. Still, hence annotations are not attached to whole Web pages, rather to reusable document components, their remarks to the proof can be exchanged between each other.

5 Document Generation

In this section the process of document generation is outlined. The goal of this process is the dynamic creation of personalized Web documents adapted to special user and client requirements. Firstly, the overall architecture of document generation via the pipeline concept is explained. After that important performance issues, especially possible caching scenarios are described.

5.1 Pipeline Concept

The process of document generation is based on a pipeline concept allowing the stepwise transformation from document components to the generated output. The advantage of partitioning the transformation process into several steps is to achieve code reuse and higher performance by effective caching mechanisms.

In the beginning, an XML-document according to the end user's request is retrieved from a component repository. This document is an instance of the XML-schema grammar forming the basis of the document model and represents an abstract description of the Web content to be presented. This abstract description contains not only information about subcomponents to be included but also metainformation concerning adaptation rules. Generally, all possible presentation forms of the component concerning content, layout, and structure are encapsulated in this document.

Depending on the current user model - which in this case comprises both the user's semantic preferences and the description of technical client preferences - this document is subdued to a series of transformations, each implemented by a specific XML transformer. In each transformation step a certain aspect of adaptation is considered and corresponding conversions to the document are performed. These may comprise selection and configuration of components and/or their subcomponents. Finally, the last transformation step in the pipeline generates a Web document in a specific output format, e.g. HTML, WML or cHTML. The pipeline-based architecture of the document generator is depicted in Figure 3.

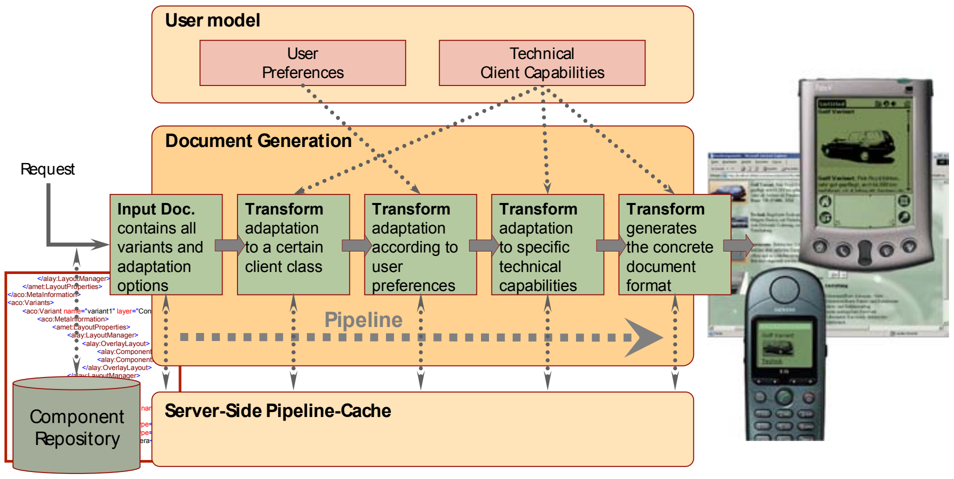
User model
User Preferences
Technical Client Capabilities
Document Generation
Request
Input Doc. contains all variants and adaptation options
Transform adaptation to a certain client class
Transform adaptation according to user preferences
Transform adaptation to specific technical capabilities
Transform generates the concrete document format
Pipeline
Component Repository
Server-Side Pipeline-Cache

Figure 3 Pipeline-based Document Generation

In order to examine the stepwise document generation process an example scenario for product presentations is being designed. In this scenario the user gets different automobiles presented according to his personal preferences and/or client capabilities. The prototype addresses all aspects of adaptation mentioned in Section 4.2., and is realized on the basis of the pipeline-based publishing framework Cocoon [27].

The transformation pipeline of this scenario consists of four steps. Firstly, the presentation is adapted to a specific client class, such as a PDA or a desktop PC. While a Web page for a desktop computer contains four document components such as page header, navigation pane, product visualization, and product description; a presentation for a PDA only contains a grayscale image (product visualization) and some textual explanations (product description). Secondly, personal user preferences are taken into account. Users interested in technical details are presented a smaller image and an in-depth textual product description, "everyday users" are rather presented visual information e.g. more images or a video clip. (It would also be possible to perform user adaptation in two steps, namely adaptation to a given user class followed by adaptation to the specific user. However we abstained from this separation in this scenario.) Personal annotations created by users are included in this second transformation step, too. Thirdly, adaptation according to concrete technical capabilities is performed. According to different display capabilities and bandwidth ratios, the size of images or the bitrate of videos to be included is varied dynamically. As the fourth step, the final presentation in a concrete output format is generated.

One very important aspect of pipeline design is the proper arrangement of transformation blocks in order to support caching by reusing partially adapted documents. It is very useful to perform the adaptation to the client class firstly, still before other adaptation processes. Since the number of possible client classes is significantly less than the number of conceivable user preferences, the output of the first transformer can be effectively cached and reused for a number of requests. Similarly, the output of the second transformer can be cached and reused for different instances of a client class with varying technical parameters.

The following subsection deals with further performance issues regarding the document model and the generation process.

5.2 Performance Issues and Caching

One of the biggest challenges of managing adaptive, personalized Web presentations is the performance of content delivery. Effective Web content management requires Web pages to be generated dynamically and thus can cause significant server load. One approach to address this bottleneck is the deployment of server-side caches [9] to store dynamically generated content temporarily.

Whereas there exists a number of approved mechanisms for caching static content [40], there are barely approaches to accelerate dynamic applications. The decision, which content pieces can be cached and when they have to be updated or deleted, presumes proper knowledge of the structure of and the data dependencies within Web applications. Thus it is important to make provisions for the caching behavior of dynamic content already during the authoring of Web presentations.

A widely used technique for efficiently caching dynamic Web pages is their decomposition into fine-granular fragments with different cache properties. This concept was successfully implemented in [8] and is also the basis of the ESI-specification (Edge Side Includes) [2].

The proposed document model allows defining content fragments with independent cache properties on different abstraction levels. It is possible to attach cache metadata to components in order to control server-side caches. The meta attribute *dynamism* determines whether the given component contains static or dynamic content. Static components can be stored in the server cache automatically. For dynamic components the second meta attribute *expirationTime* specifies the time after the corresponding cache entry should be updated. Figure 4 illustrates this mechanism by the example of the title page of an online newspaper. This contains a number of columns with different expiration times. While the current weather report has to be updated every 15 minutes, the static "overview" does not change temporally. By modeling such Web pages as compositions of document components with varying cache properties the server-side cache component of the document generation architecture can be controlled intelligently.

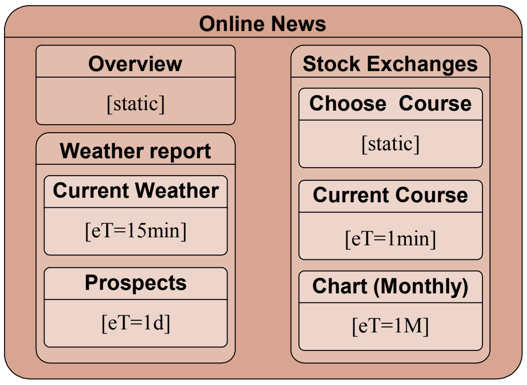

Figure 4. Caching Properties Attached to Web Components (eT=Expiration Time)

Another important performance aspect is the caching of partially adapted Web components along the pipeline. In order to achieve this, the output of each transformer can be cached temporarily. The output of a transformer is unequivocally determined by the input document, the used adaptation method, and a number of transformation parameters acquired from the user model. As long as these parameters are the same for a number of subsequent requests, the output of the transformer can be cached. Of course the dependencies between serially switched transformers also have to be considered. In the case when the stylesheet of a certain transformer or a certain input document containing dynamically generated data is changed, the caches of all affected transformers in the subsequent pipeline have to be updated. Figure 3 depicts the adjustment of the server-side cache in the pipeline architecture.

6 Conclusions and Future Work

This paper introduces a component-based document model for dynamic, adaptive Web applications. The model aims at supporting conceptual design of Web sites and effective page generation by defining fine-granular reusable, adaptable Web components on different abstraction levels, namely media components, content units, document components, and the hyperlink level. Model benefits, such as component reuse, support for adaptation, and collaboration by annotations were outlined. Finally, the process of document generation and corresponding performance issues were treated.

The major characteristics of the described approach are being verified by a first implementation of a prototype.

Future work includes the refinement of the XML-schemas. A suitable representation form for technical information on clients and user models is to be specified. A detailed performance analysis of the pipeline-based document generation and the development of algorithms for effective server-side cache control based on the suggested metadata model are also to be done.

Another main activity will focus on the authoring process of adaptive Web components. Existing design and process models for hypermedia application development are to be analyzed regarding their extensibility for adaptive component-based Web sites. Furthermore, a component repository for creating, storing, retrieving, configuring, and publishing of adaptive content components needs to be designed and implemented.

References

1. iMarkup Product Homepage. http://www.imarkup.com.
2. Edge Side Includes (ESI) Overview. Oracle Corporation, Akamai Technologies Inc. http://www.esi.org, 2000.
3. Wireless Application Group: User Agent Profile Specification, WAP Forum. 2001.
4. Bra, P. D., Houben, G.-J. and Wu, H., AHAM: A Dexter-Based Reference Model for Adaptive Hypermedia. in HYPERTEXT '99, 10th ACM Conference on Hypertext and Hypermedia: Returning to Our Diverse Roots, Darmstadt, Germany, 1999.
5. Brusilovsky, P. Methodes and Techniques of Adaptive Hypermedia. *User Modeling and Adapted Interaction*, vol. 6., pp. 87-129, 1996.
6. Brusilovsky, P. Adaptive Hypermedia. *User Modeling and User Adapted Interaction*, vol. 11, pp. 87-110, 2001.
7. Ceri, S., Fraternali, P. and Bongio, A., Web Modeling Language (WebML): a modeling language for designing Web sites. in 9th International Conference on the WWW (WWW9), Amsterdam, 2000.
8. Challenger, J., Iyengar, A. and Witting, K., A publishing system for efficiently creating dynamic web data. in IEEE INFOCOM 2000, 2000.
9. Challenger, J., Iyengar, A., Dantzig, P., Dias, D. and Mills, N., Engineering Highly Accessed Web Sites for Performance. in *Lecture Notes in Computer Science - Web Engineering*: Springer-Verlag Berlin Heidelberg, 2001.
10. De Bra, P. and Ruiter, J. P., AHA! Adaptive Hypermedia for All. in WebNet, 2001.
11. De Bra, P., Aerts, A., Smits, D. and Stash, N., AHA! Version 2.0, More Adaption Flexibility for Authors. in AACE ELearn'2002, 2002.
12. Denoue, L., Adding Metadata to improve retrieval: Yet Another Web Annotation System. TR1999-01, February 1999.
13. Denoue, L. and Vignollet, L., An annotation tool for Web browsers and its applications to information retrieval. in RIAO2000, Paris, 2000.
14. Fiala, Z. and Meissner, K., Annotating Virtual Web Documents with DynamicMarks. in XSW2003 XML Technologien für das Semantic Web, Berlin, Germany, 2003.
15. Frasincar, F., Houben, G. and Vdovjak, R., Specification Framework for Engineering Adaptive Web Applications. in Tenth International World Wide Web Conference (WWW11), 2001.
16. Gaedke, M., Rehse, J. and Graef, G., A Repository to facilitate Reuse in Component-Based Web-Engineering. in International Workshop on Web Engineering at the 8th International World-Wide Web Conference (WWW8), Toronto, Ontario, Canada, 1999.
17. Gaedke, M. and Graef, G., WebComposition Process Model: Ein Vorgehensmodell zur Entwicklung und Evolution von Web-Anwendungen. in 2. Workshop Komponentenorientierte betriebliche Anwendungssysteme (WKBA2), Wien, Austria, 2000.
18. Gaedke, M., Segor, C. and Gellersen, H.-W., WCML: Paving the Way for Reuse in Object-Oriented Web Engineering. in ACM Symposium on Applied Computing (SAC2000), Villa Olmo, Como, Italy, 2000.
19. Geyer-Schulz, A., Koch, A. and Schneider, G., Virtual Notes; Annotations on the WWW for Learning Environments. in AMCIS 1999, 1999.
20. Graef, G. and Gaedke, M., Construction of Adaptive Web-Applications from Reusable Components. in 1st International Conference on Electronic Commerce and Web Technologies (EC-Web 2000), London, UK, 2000.

21. Halasz, F. and Schwartz, M. The Dexter Hypertext Reference Model. *Communications of the ACM*, vol. 37, pp. 30-39, 1994.
22. Healey, J., Hosn, R. and Maes, S., Adaptive Content for Device Independent Multi-modal Browser Applications. in Adaptive Hypermedia and Adaptive Web-Based Systems - Second International Conference, AH2002, Malaga, Spain, 2000.
23. Isakowitz, T., Stohr, E. A. and Balasubramanian, P. RMM: A Methodology for Structured Hypermedia Design. *Communications of the ACM*, 1995.
24. Jablonski, S. and Meiler, C. Web-Content-Managementsysteme. *Informatik Spektrum*, vol. 25/2, 2002.
25. Jörding, T., Temporary User Modeling for Adaptive Product Presentations in the Web. in Seventh International Conference on User Modeling, Banff, Canada, 1999.
26. Klyne, G., Reynolds, F., Woodrow, C., Ohto, H. and Butler, M. H., Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies. W3C Working Draft http://www.w3.org/TR/2002/WD-CCPP-struct-vocab-20021108/, 08 November 2002.
27. Langham, M. and Ziegeler, C., *Cocoon: Building XML Applications*: New Riders, 2002.
28. Laroussi, M., Ben Ahmed, M. and Marinilli, M., An Adaptive Document Generation Based on Matrix of Contents. in Adaptive Hypermedia and Adaptive Web-Based Systems - International Conference, AH 2000, Trento, Italy, 2000.
29. Lassila, O. and Swick, R. R., Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, 1999.
30. Lyman, P., Varian, H. R., J., D., Strygin, A. and Swearingen, K., How much Information? University of California at Berkeley 2000.
31. Martinez, J. M. Overview of the MPEG7 Standard (Version 7.0).
32. Ovsiannikov, I. A., Arbib, M. and McNeill, T. Annotation Technology. *International Journal of Human-Computer Studies*, vol. 50(4), pp. 329-362, 2000.
33. Röscheisen, M., Morgensen, C. and Winograd, T., Shared Web Annotations as a Platform for Third-Party Value-Added, Information Providers. Architecture, Protocols and Usage Examples. Stanford University, Technical Report CSDTR/DLTR 1995.
34. Rossi, G., Schwabe, D. and Guimarães, R. M., Designing Personalized Web Applications. in Tenth International Conference on the World Wide Web (WWW10), Hong Kong, 2001.
35. Schraefel, M., ConTexts: Adaptable Hypermedia. in Adaptive Hypermedia and Adaptive Web-Based Systems - International Conference, AH2000, Trento, Italy, 2000.
36. Schwabe, D., Rossi, G. and Barbosa, S. D. J., Systematic Hypermedia Application Design with OOHDM. in UK Conference on Hypertext, 1996.
37. Szyperski, C., *Component-Software: beyond Object-Oriented Programming*, Addison Wesley Publishing Company ed, 1997.
38. Wadge, W. and 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.
39. Wehner, F. and Lorz, A., Developing Modular and Adaptable Courseware Using TeachML. in ED-MEDIA, World Conference on Educational Multimedia, Hypermedia and Telecommunications, Tampere, Finland, 2001.
40. Wessel, D., *Web Caching*, O'Reilly ed, 2001.
41. Yea, K.-P., Crit-Link: Better Hyperlinks for the WWW. http://crit.org/~ping/ht98.html, 1998.