Personalization-based Optimization of Web Interfaces for Mobile Devices

Michael Hinz, Zoltán Fiala, Frank Wehner

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

Abstract. Developing personalized applications for the ubiquitous Web assumes to provide different user interfaces addressing heterogeneous capabilities of device classes. Major problems are the lack of sufficient presentation space and the diversity of interaction techniques, both requiring adaptive intelligent user interfaces. To meet this challenge this paper introduces an approach for the personalization-based optimization of Web interfaces for mobile devices. On the basis of a user model different adaptation issues are discussed. Firstly, static adaptation mechanisms affecting the structure of Web documents as well as layout managers enabling a device independent definition of Web presentations for heterogeneous devices are introduced. Then an interactive mechanism for dynamically predicting user preferences for hiding unnecessary information through content adaptation is presented. As a proof of concept an architecture realized by a pipeline-based document generator was developed for static/dynamic adaptation, which is partly explained in this paper.

1 Introduction

Providing personalized information becomes a significant challenge of today's Web development. The raising number of users with an increasing variety of mobile devices requires the creation and publication of content customized for different user preferences and platforms. A major problem is the diversity of display capabilities and interaction techniques provided by mobile clients, which establishes the need for adaptive intelligent user interfaces that automatically adjust their content to those heterogeneous requirements. However, existing document formats (such as HTML, cHTML or WML) are hardly suitable for engineering personalized ubiquitous Web applications, as they do not provide mechanisms for describing the adaptive behavior of content pieces in a generic way.

Existing approaches for displaying Web content on mobile devices mostly focus on restructuring or clipping existing pages according to static guidelines [1], [2], [4]. However, including the user's changing interests in this process enables not only a better personalization but also an optimized utilization of the available presentation area.

The paper is structured as follows. After addressing related work in Section 2 a short overview of our component-based document model for personalized ubiquitous Web presentations is given. Section 4 deals with different aspects of adaptation supported by the document format, and gives a short introduction to the user model. On this basis static adaptation in dependency of user and device properties, dynamic adaptation in dependency of user preferences and an automatic layout adjustment mechanism are discussed. The implemented system architecture is explained in Section 5. Section 6 concludes the paper and suggests future research directions.

2 Related Work

Recently, different solutions for adapting Web presentations and applications to mobile devices have emerged. Basically two main approaches can be distinguished. The first one adjusts existing Web pages (mostly HTML) to the limited display and interaction capabilities offered by mobile devices. The second one aims at building personalized ubiquitous Web applications “from scratch” and considers device (and user) adaptation already during the specification and implementation process.

Different mechanisms for automatically adjusting existing desktop Web pages to mobile browsers have been developed. Some solutions, e.g. Microsoft’s Pocket Internet Explorer [1] or Opera for Smartphones/PDAs [2] resize large Web pages to fit into the small displays of mobile clients. Even though all information from the original page is displayed, it is reformatted in order to eliminate horizontal scrolling. The disadvantage of this approach is a presentation often featured with unnecessary information or layout fragments. Therefore, Web clipping techniques have emerged which firstly analyze the structure of Web pages. By discovering priorities, page fragments are classified as either important or unimportant, and the latter are excluded from the “clipped” presentation. Two strategies for defining priorities exist. The first one uses intelligent algorithms to automatically classify page fragments [3], [4]. The second strategy [5], [6] requires a manual definition of priorities. As further interesting approaches we mention HANd [7] and SmartView [8] which structure the original Web page into zones. Through automatically generated summary pages or thumbnails every zone can be reached via navigation. The advantage of those techniques is that no information is clipped since by navigation every zone can be reached. Still, extra navigation is required and by splitting a page the overview gets lost. Therefore the user’s mental load rises. A similar approach for text browsing [9] enables the summarization of texts with an "accordion" display technique.

The main advantage of the approaches mentioned above is that they are principally suitable for adapting arbitrary Web pages. However, evaluations ([10]) show that it is often impossible to predict (or enforce) the result of the transformation process and that in many cases erroneous output pages are provided. Furthermore, since all these approaches operate on the HTML-based presentation view of their input pages, adaptation is restricted to the exclusion or rearrangement of content pieces. On the other hand, we claim that effective device adaptation has to be already considered during the conceptual and navigational design of Web applications.

Recently, different approaches for modeling and engineering ubiquitous personalized Web systems have emerged. Among the most significant ones we mention WebML [11] and Hera [12]. However, all these approaches focus on the conceptual modeling and design of hypermedia applications, not supporting the flexible reuse of adaptable implementation artifacts. Furthermore, device adaptation is not a central aspect of these approaches. To fill this gap, the project AMACONT [13] recently introduced a component-based document format for personalized ubiquitous Web presentations [14]. It focuses not on the conceptual design of Web applications, but on the challenge to reuse adaptable implementation artifacts. In this paper a detailed overview of personalization issues (with a special focus on device adaptation) is given.

3 The Document Model

In the Amacont approach Web sites are composed of configurable Web components [14]. These components are instances of an XML grammar representing adaptable content on different abstraction levels. Web sites are constructed by aggregating and linking components to complex document structures. During Web page generation these abstract document structures are translated into Web pages in a concrete output format, adapted to a specific user model or client device, respectively.

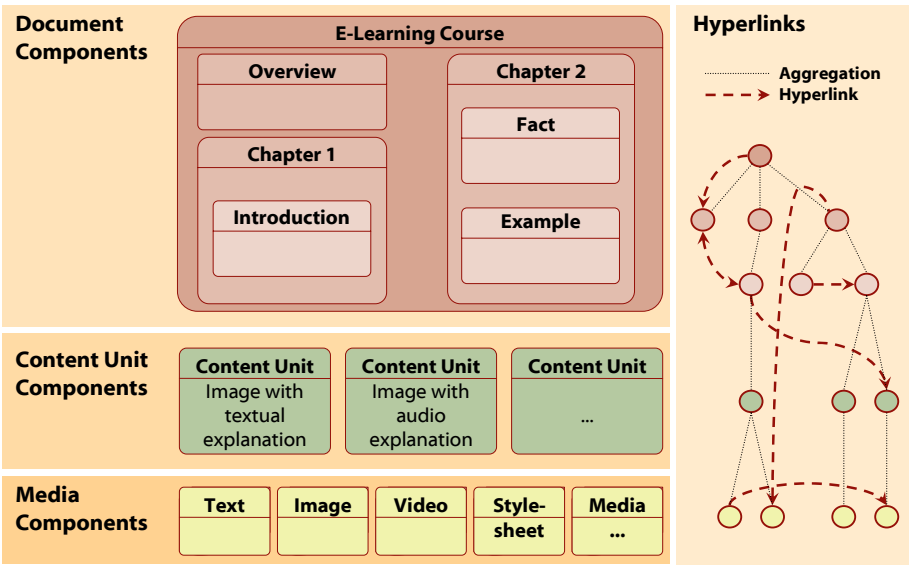

Fig. 1. The document model

The lowest abstraction level introduces *media components* that encapsulate concrete media assets. These comprise text, structured text (e.g. HTML), images, sound, video, Java applets and may be extended arbitrarily. Besides MPEG7-based technical properties 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 unit components*. 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 (Section 4.3).

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. For a detailed introduction to the document model the reader is referred to [14].

4 Adaptation Support

The component-based document format aims at supporting adaptation by two mechanisms [15]. 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. Both adaptation aspects can be declared by attaching specific adaptation metadata to components. During document generation, this metadata is evaluated according to an XML-based user model and the corresponding adaptation processes are performed.

Furthermore, two types of adaptation or personalization can be 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 [16]. 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.

This section provides an overview of AMACONT's versatile adaptation capabilities. Firstly, the structure of the user model is depicted which is used across all examples. Then, different aspects of static and dynamic personalization are described in detail. All introduced adaptation examples aim at optimizing Web presentations to mobile end devices.

4.1 The User Model

The adaptation of components happens according to an XML-based user model. This is composed of a number of profiles that can be seen in Fig. 2 Each profile relies on

CC/PP (Composite Capability / Preference Profiles), an RDF grammar for describing device capabilities and user preferences in a standardized way [17]. However, as being a general grammar, CC/PP makes no assumptions on concrete resource characteristics. Therefore, an XML-based schema was developed for each profile. By adding new profiles the user model can be extended arbitrarily.

The first part (*IdentificationProfile*) of the user model contains information to identify users. Besides a set of general properties (name, email etc.), arbitrary extensions are allowed. Technical properties and capabilities of users' client devices are stored in *DeviceProfile*. It is represented on the basis of the WAP User Agent Profile (UAProf [18]) providing a common vocabulary for WAP devices. To support also other mobile devices (e.g. PDAs), specific extensions of UAProf have been made. Furthermore, as usually there are much more users than devices, it is also possible to reference separately stored device profiles.

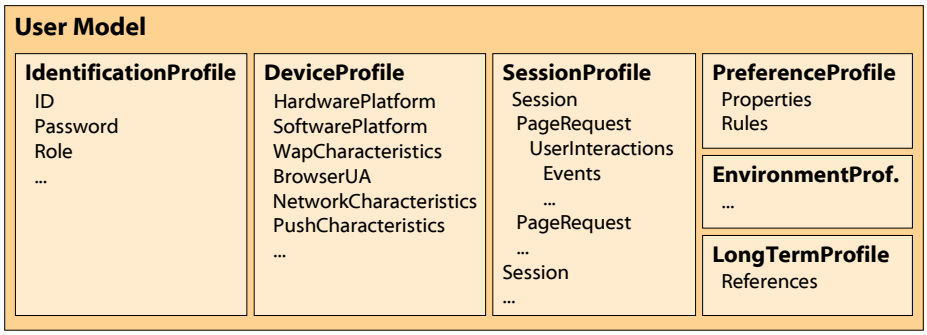

Fig. 2. The user model

The *SessionProfile* integrates user interactions by grouping them to page requests and sessions. It stores past user interactions in the form of events related to data acquisition objects (see section 4.4). Based on this interaction history list the user modeling process generates new knowledge about the user in term of rules (see Section 4.4). Those rules are stored in the *PreferenceProfile* and used by the document generator to adapt the content of a web page to user preferences. The last two profiles are placeholders for upcoming research. *EnvironmentProfile* will provide information about the context and location of the user for supporting location based services. *LongtermProfile* will have a bridging function between a special user model and comprehensive models containing information about all users of the system. E.g. the user class membership of a user will be represented by this profile in order to reduce server load by handling groups of users together.

4.2 Static Adaptation in Dependency of User and Device Properties

The document format described in Section 3 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.

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 properties. 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 video tapes, different content depending on the bandwidth of the user's device is presented.

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

```
<AmaDocumentComponent name="Film">
 <MetaInformation>
      ...
 </MetaInformation>
 <Variants>
      <Variant name="Video_Trailer">
           ...
      </Variant>
      <Variant name="Cover_Picture">
           ...
      </Variant>
 </Variants>
</AmaDocumentComponent>
```

```
<AdaptiveProperties>
  <If>
 <Expr operator="greaterThan">
      <UserModelParam>
           Bandwidth
      </UserModelParam>
      <Const>64000</Const>
 </Expr>
      <Then res="Video_Trailer"/>
      <Else  res="Cover_Picture"/>
 </If>
<AdaptiveProperties>
```

The processing XSLT style sheet substitutes the integer variable "Bandwidth" by its value from the current user model, performs the selection method and determines the proper variant of the "Film" 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 act as an interpreter for this "selection method language".

4.3 Automatic Layout Adaptation

In order to describe the presentation of component-based Web documents, AMACONT allows attaching XML-based layout descriptions to components. Inspired by the layout manager mechanism of the Java language (AWT and Swing) and the abstract user interface representations of UIML [19] and XIML [20], they describe a client-independent layout allow abstracting from the exact resolution of the display or the browser's window. Note that layout managers of a given component only describe the presentation of its immediate subcomponents, which encapsulate their own layout information in a component-based way.

At current time four layout managers can be 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. *GridTableLayout* enables to lay out components in a grid with a configurable number of columns and rows. Finally, *OverlayLayout* allows to present components on top of each other.

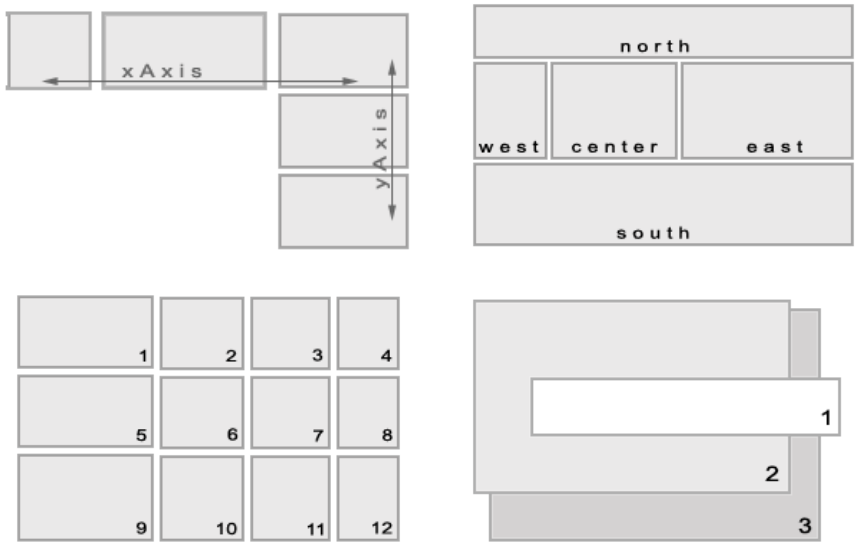

Fig. 3. Layout managers: upper left: BoxLayout, upper right BorderLayout, lower left: GridTableLayout, lower right OverlayLayout

Layout managers are formalized as XML elements with specific attributes. Two kinds of attributes exist: *layout attributes* and *subcomponent attributes*. Layout attributes declare properties concerning the overall layout and are defined in the corresponding layout tags. As an example the *axis* attribute of BoxLayout determines whether it is laid out horizontally or vertically. On the other hand, subcomponent attributes describe how each referenced subcomponent has to be arranged in its surrounding layout. Table 2 summarizes the possible attributes of BoxLayout by describing their names, role, usage (required or optional) and possible values.

Table 2. Example: layout attributes of the BoxLayout manager

Layout Attributes	Meaning	Usage	Values
axis	orientation of the BoxLayout	req.	xAxis | yAxis
space	space between subcomponents	opt.	percent or absolute
width	width of the whole layout	opt.	percent or absolute
height	height of the whole layout	opt.	percent or absolute
border	width of border between subcomp.	opt.	percent or absolute
Subcomponent Attributes			
align	horizontal alignment of subcomp.	opt.	left | center | right
valign	vertical alignment of subcomponent	opt.	top | center | bottom
ratio	space taken by subcomponent	opt.	percent
wml_visible	show on same WML card?	opt.	boolean
wml_desc	link description for WML	opt.	string

The optional attribute *wml_visible* determines whether in a WML presentation the given subcomponent should be shown on the same card. If not, it is put onto a separate card that is accessible by an automatically generated hyperlink, the anchor text of which is defined in *wml_description*. This mechanism of content separation and navigation adaptation is used since the displays of WAP capable mobile phones are very small.

The exact rendering of media objects happens during document generation time by XSLT stylesheets that transform components with such abstract layout properties to specific output formats. Three stylesheets for converting those descriptions to XHTML, cHTML and WML output have been realized.

4.4 Dynamic Adaptation Issues

The mechanisms described above support adaptability by adjusting Web presentations to (mostly) static user and device properties. However, in order to realize dynamic adaptation (or adaptivity), they have to be extended by additional feedback mechanisms. User interactions have to be captured on the client and sent back to the server in order to update the user's preference profile, i.e. to automatically generate adaptation rules according to the user's browsing behavior. In contrast to other approaches (e.g. [3], [5], [6]), this allows to adjust Web presentations to even dynamically changing user interests.

Note that this strategy can be effectively used for optimizing Web pages on mobile devices with limited presentation space. As an example, take the case of an interactive multimedia Web presentation allowing to perform interactions on selected media items. A user being more interested in textual information (due to the limited display capabilities of his browser) could collapse images and enlarge texts. A corresponding learning algorithm could recognize this and generate the appropriate adaptation rules which automatically collapse all images for the user's display.

A further possibility is to provide observed media components with a special semantic meaning in order to predict semantic user preferences. Let us take the case of an online product presentation where a user enlarges a picture containing technical features of a selected product and then changes to the next product. The system could

establish a rule that the user is interested in technical details and generate the next product presentation according to this rule.

Acquire Interactions

In order to observe users' browsing behavior, our developed system allows to track interactions that are performed on media components included in a Web page. During server side document generation specific code fragments (implemented as JavaScript or JScript functions) are embedded and configured for each media component to be observed. They allow capturing user interactions on the client side and sending them back to the server, where they are stored in history lists (session profile). Acquirable interactions are listed in Table 3.

Table 3. Acquirable interactions of observed components

observed component	**acquirable interactions**
video and audio component	started, paused at, stopped at
image component	minimized, maximized, printed
scroll text component	scrolling time, end reached
toggle text component	enlarged, collapsed
pop up text component	pop up

In order to make media components observable, component authors have to provide them with specific metadata. Hence, semantic metadata in the form of attribute-value pairs (e.g. content="technical details") can be attached to them. Thus, the semantic preferences of user's interacting with those objects can be predicted.

Processing Interactions

By evaluating interactions, suggestions on users' preferences and knowledge can be made and parts of the user model can be updated or specialized. In our developed prototype application focusing on product presentation this specialization is performed by the incremental learning algorithm CDL4 (Complementary Discrimination Learning [21]). The algorithm was approved as very useful in adaptive multimedia product presentations in an earlier project of the authors' research group [22].

CDL4 utilizes decision lists in order to describe user models. A decision list is a series of simple rules describing user preferences. As an example, the following decision list claims that the user is not interested in multimedia information about actors other than the main actor:

$$[((actor \neq mainActor) \wedge (medium \neq text) \rightarrow noInterest),$$
$$(default \rightarrow interest)]$$

If no rules from earlier sessions exist, CDL4 starts with a minimal default decision list (see second line in the example above) in the beginning of each user session. According to the user's interaction behavior, this is extended (specialized) in an incremental way.

Interactions stored in the session profile are transformed to so called training instances. Training instances are also formed as single decision rules and serve as the input for the CDL4 algorithm. For instance, if the user enlarges a picture component containing the biography of a supporting movie actor, the server generates following training instance:

[biography, supportingActor, picture → interest]

Each time a new training instance is provided, the algorithm has to check whether its current decision list already covers this new instance. If yes, the decision list remains unchanged. Otherwise, the algorithm learns this new instance and updates (specializes) the corresponding decision list by changing an existing rule or inserting a new one. In our example, the update decision list would look like this:

[((actor ≠ mainActor) Λ (medium ≠ text) Λ (medium ≠ picture) → noInterest), (default → interest)]

At the user's next document request, the inserted media components are configured according to the new rules. For more details on CDL4 the reader is referred to [21].

5 Generating Adaptive Web Documents

Document generation aims at transforming complex component structures to Web pages adapted to user properties and preferences as well as device profiles. It is performed in a stepwise, pipeline oriented way (Fig. 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 (containing also the device profile), 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 4.2).

Fig. 4 shows a possible scenario with three steps, namely adaptation to a certain client class (e.g. PDA, cell phone or notebook), then to static user properties (age, gender, knowledge level, etc.) and finally to semantic user preferences (e.g. interests, media preferences).

In this scenario the first two adaptation steps are performed according to the variant selection mechanism described in Section 4.2. Thus, the hierarchy of components is adjusted to static user properties and device profiles.

The third step, namely dynamic adaptation according to changing user preferences affects not the aggregation hierarchy of the overall component structure but the presentation parameters of single media components. For example, an image can be inserted minimized or maximized, a text can be presented in a short or in a long form, or even videos can be started automatically. These decisions are made by the CDL4-algorithm according to the rules stored in the preference profile.

After the component hierarchy to be presented and the parameters of media objects have been determined, the resulting adapted document has to be transformed to a

specific output format (XHTML, cHTML, WML etc.). According to the layout managers described in Section 4.3, this rendering happens automatically. Moreover the data acquisition objects for tracking user interactions are included in this transformation step, too. Again, they enable to track user interactions in the newly generated presentation. This loop enables a dynamically adaptation process with an always up-to-date user model.

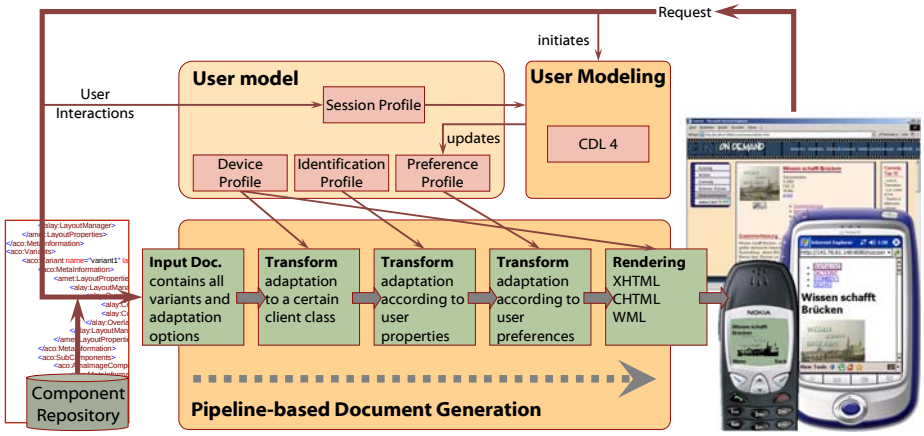

Fig. 4. Pipeline-based document generation

6 Conclusion and Future Work

In this paper an overview of the adaptation issues provided by the XML-based document model and the system architecture of the AMACONT project was given. Both static adaptation issues based on user and device properties and dynamic personalization aspects according to dynamically changing user preferences were discussed. Furthermore, a pipeline-based document generator was introduced for performing those adaptations in a stepwise way. We have shown how the Web interface of mobile devices can be optimized by those personalization techniques. Especially the observation of users and the prediction of their preferences enabled an automatic prioritization of content and therefore the hiding of unnecessary information from the user.

Future work concentrates on the authoring process of dynamically personalized Web documents for heterogeneous mobile devices. A modular framework for creating and configuring components in different stages of the authoring process is being built. Furthermore, performance aspects of the system architecture will be addressed, too. Since dynamic adaptation mechanisms cause significant server load, optimizing the performance seems to be an important effort when handling lots of users. Initial tests showed that the number of requests and the structure of existing rules play an important role when the system manages dynamic adaptation. Reducing rules representing user preferences to a minimum could improve the overall performance.

References

1. Pocket IE: http://www.microsoft.com/windowsmobile/products/pocketpc/default.mspx
2. Opera's Small Screen Rendering: http://www.opera.com/products/smartphone/smallscreen
3. Gomes, P.; Tostao, S.; Goncalives, D.; Jorge, J.: Web Clipping: Compression Heuristics for Displaying Text on a PDA. In Proceedings of the Mobile HCI'01, 2001.
4. Bickmore, T.; Schilit, B.: Digestor: Device-independent Access to the WWW. In Proceedings of the WWW6 conference, 1997.
5. Hori, M.; Kondoh, G.; Ono, K.; Hirose, S.; Singhai, S.: Annotation-based Web Content Transcoding. In Proceedings of the WWW9 conference, Amsterdam (Netherlands), 2000.
6. Palm Web Clipping: http://www.palmos.com/dev/tech/webclipping/resources.html
7. González-Castano, F.; Anido Riffón, L.; Costa Montene-gro, E.: A New Transcoding Technique for PDA Browsers, Based on Content Hierarchy. In Proceedings of the 4th International Symposium on Mobile HCI, 2002.
8. Milic-Frayling, N.; Sommerer, R.: SmartView: Flexible viewing of web page contents. Poster presentation at the WWW11 conference, 2002.
9. Buyukkokten, O.; Garcia-Molina, H.; Paepcke, A.: Seeing the whole in parts: Text summarization for web browsing on handheld devices. In Proceedings of the WWW10 conference, Hong-Kong, 2001.
10. Chen, Y.; Ma, W.; Zhang, H.: Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices. In Proceedings of the WWW12 conference, 2003.
11. Ceri, S.; Fraternali, P.; Bongio, A.; Brambilla, M.; Comai, S.; Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann, ISBN 1558608435, 2003.
12. Vdovjak, R.; Frasincar, F.; Houben, G.J.; Barna, P.: Engineering Semantic Web Information Systems in Hera. In Journal of Web Engineering, Vol.2 No.1&2, RP, 2003.
13. AMACONT Project: http://www-mmt.inf.tu-dresden.de/english/Projekte/AMACONT/
14. Fiala, Z.; Hinz, M.; Meißner, K.; Wehner, F.: A Component-based Approach for Adaptive, Dynamic Web Documents. In Journal of Web Engineering, Vol.2 No.1&2, RP, 2003.
15. Fiala, Z.; Hinz, M.; Wehner, F.: An XML-based component architecture for personalized adaptive web applications. In Workshop Personalisierung mittels XML-Technologien, Berliner XML Tage, Pages 370 - 378, 2003.
16. Fiala, Z.; Hinz, M.; Houben, G.; Frasincar, F.: Design and Implementation of Component-based Adaptive Web Presentations. In Proceedings of the 19th Annual ACM Symposium on Applied Computing, 2004.
17. Klyne, G.; Reynolds, F.; Woodrow, C.; Ohto, H.; Hjelm, J.; Butler, M.; Tran, L.: Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation 15 January 2004.
    http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/
18. Wireless Application Group: User Agent Profile Specification. Open Mobile Alliance WAP Forum 2001.
19. User Interface Markup Language (UIML): Specification Draft Language, Version 3.0, February 2002. http://www.uiml.org/specs/
20. Puerta, A; Eisenstein, J.: XIML: A Universal Language for User Interfaces. In Proceedings of the Conference on Intelligent User Interfaces, 2002.
21. Shen, W.: An efficient Algorithm for Incremental Learning of Decision Lists. Technical Report, USC-ISI-96-012, University of Southern California, 1996.
22. Jörding, T., Meissner, K.: Intelligent Multimedia Presentations in the Web: Fun without Annoyance. In Proceedings of the WWW7 conference, Brisbane, 1998.