Engineering the Presentation Layer of Adaptable Web Information Systems

Zoltán Fiala[1], Flavius Frasincar[2], Michael Hinz[1],
Geert-Jan Houben[2], Peter Barna[2], Klaus Meissner[1]

[1] Technische Universität Dresden
Mommsenstr. 13, D-01062, Dresden, Germany
{zoltan.fiala,mh5,kmeiss}@inf.tu-dresden.de
[2] Technische Universiteit Eindhoven
PO Box 513, NL-5600 MB, Eindhoven, The Netherlands
{flaviusf,houben,pbarna}@win.tue.nl

Abstract. Engineering adaptable Web Information Systems (WIS) requires systematic design models and specification frameworks. A complete model-driven methodology like Hera distinguishes between the conceptual, navigational, and presentational aspects of WIS design and identifies different adaptation "hot-spots" in each design step. This paper concentrates on adaptation in the presentation layer and combines the modeling power of Hera with the versatile presentation capabilities of the AMACONT project. After discussing different aspects of presentation layer adaptation, the layout manager mechanism of AMACONT for describing the adaptable layout of ubiquitous Web presentations is introduced. Then the RDFS-based Hera schema for presentation models is presented, allowing to assign AMACONT layout descriptors to Hera slices. According to this formalization, Hera application model instances are automatically transformed to a component-based AMACONT implementation that can be adjusted to different end devices and output formats. The XML-based transformation process is explained in detail, and the resulting methodology is exemplified by a prototype application.

1 Introduction

Engineering *personalized* Web Information Systems (WIS) is a complex process that has to be based on disciplined, systematic methodologies. Among these we distinguish the model-based methodologies due to the numerous benefits they offer: easy system analysis, modular decomposition, adaptation "hot-spots", flexibility, maintainability etc. By identifying crucial phases of Web development, such an approach helps designers and programmers to proceed in a structured way.

Hera [1] is a model-driven design methodology and specification framework focusing on the development of personalized WIS. Based on the principle of separation of concerns it distinguishes three design steps: *conceptual design, navigational design*, and *presentation design*. At each design step different aspects

of adaptation can be specified in terms of formal models [2]. In order to make the semantics of models explicit, Hera uses RDF(S) [3, 4]. RDFS inheritance mechanism proved to be useful for reusing different design artifacts [2]. However, previously Hera's presentation model has not been formalized, nor was adaptation at the presentation level implemented in the Hera tools.

The AMACONT project [5] recently introduced a *component-based XML document format*. It enables to compose personalized ubiquitous Web presentations from reusable document components encapsulating adaptive content, behavior, and layout. A special focus of AMACONT lies on the presentation layer of adaptive Web applications. It allows to describe the layout of adaptable Web components in a device independent way. Furthermore, a document generator for adjusting those components to different formats and devices has been developed.

Taking advantage of the analogies between Hera's *slices* and AMACONT's *components*, this paper aims at combining the modeling power of Hera with the presentation capabilities provided by AMACONT. The AMACONT concept of abstract *layout managers* is adopted for the presentation model of Hera, and its RDFS-based formalization is provided. According to this formalization of the presentation model, Hera specifications can now be automatically transformed to AMACONT's adaptable Web components.

The paper is structured as follows. After addressing related work and different aspects of presentation level adaptation in Section 2, a short overview of the model-driven Hera specification framework is given in Section 3. Section 4 explains the layout manager mechanism of AMACONT in detail. Finally, Section 5 introduces the formalized Hera presentation model schema and depicts the overall data transformation and presentation generation process. As a proof of concept the proposed methodology was implemented for an art museum WIS. To help the reader grasp the different models, graphical excerpts are provided for the different RDFS model representations.

2 Presentation Layer Adaptation

2.1 Presentation Design

Recently, different approaches for modeling hypermedia or Web applications have emerged. Among the most significant contributions besides Hera we mention the Object Oriented Hypermedia Design Model (OOHDM [6]), the Web Modeling Language (WebML [7]), the UML-based Web Engineering approach (UWE [8]), Object-Oriented Web-Solutions Modelling (OOWS [9]), and the Object-Oriented Hypermedia Method (OO-H [10]). Even though utilizing different formalisms and notations, all these methodologies are similar in distinguishing among (1) the *conceptual* design of the application domain, (2) the *navigational* design defining the (abstract navigational) structure of the hypermedia presentation, and (3) the *presentation* design specifying the rendering of navigation objects (layout).

Presentation design aims at declaring the "look-and-feel" of a Web application independent from its implementation. For this reason, some WIS method-

ologies use explicit models allowing to compose the layout of Web presentations from abstract user interface elements (also called *abstract data views* in OOHDM or *user interface views* in UWE). On the other hand, methods like WebML do not include a specific model for expressing presentation at the conceptual level and hide the presentation in application specific XSLT stylesheets. The drawback of the latter approach is that system maintenance becomes difficult.

As a significant approach for the automatic generation of Web-based multimedia presentations we mention the Cuypers engine [11]. Its presentation design uses Prolog rules declaring qualitative and quantitative constraints on the spatial and temporal arrangement of presentation elements. Though being very flexible, the lack of an explicit model makes it difficult to predict (or enforce) how the presentation should look like in the end of the generation process. Recently, Cuypers was enhanced by a multimedia formatting vocabulary for explicitly specifying the visual layout of multimedia presentations. Similar to the well-proven layout management of TeX, it allows to adjust presentation elements according to a top-to-bottom (`vbox`) or a left-to-right (`hbox`) order. Still, more difficult arrangements (like the layout managers `GridLayout`, `BorderLayout`, or `OverlayLayout` from the Java AWT libraries) are not supported, yet.

In recent years, different XML-compliant user interface descriptions languages, like UIML and XIML have emerged [12]. However, besides abstract layout specification they also aim at modeling other UI aspects, such as domain objects, user tasks, or dialogs. In Human-Computer Interaction research user interface design has often not been integrated with the earlier stages in WIS design. Similar to the Cuypers approach, constraint-based techniques for dynamically adjusting abstract XML presentation descriptions to the displays of mobile devices are discussed in [13].

2.2 Adaptation in Presentation Design

As *personalization* and *adaptation* become prominent issues of WIS design, we claim that it is inevitable to include adaptation aspects in presentation design. Still, whereas adaptation has been extensively considered in conceptual and navigational design [2, 7], adaptation in the presentation layer has not yet been a central issue of WIS methodologies. Nevertheless, in order to support users' different layout preferences and client devices, different adaptation aspects have to be specified in presentation design.

- Firstly, it is important to adjust *media instances* to varying technical system parameters provided by different network environments and client devices, such as bandwidth, display resolution, or color depth. In order to consider such differences, variants of selected media items have to be provided with different quality.
- A further adaptation target is the corporate design (the "look-and-feel") of the Web presentation. As an example, in an online shop it is common to provide different design variants according to different user properties (age, education, interests, visual impairments etc.) but also according to

external parameters (seasons, events, anniversaries etc.). Depending on these aspects, design elements such as background colors, fonts (size, color, type), or buttons can be varied.

– Finally, the *spatial* and *temporal* adjustment of layout elements is also an important personalization issue. Depending on the screen size, the supported document formats, and the interaction techniques provided by different client devices, the presentation components should be displayed accordingly. We mention 3 mechanisms for display adaptation: (1) *Reorganization*: In this case the arrangement of media elements on a Web page is adapted. Whereas for example the tabular arrangement of layout components may look good on conventional desktops, it could cause a lot of undesirable horizontal scrolling when being browsed on handhelds with limited display size. (2) *Exclusion*: Information being unsuitable for a particular browser (e.g. a picture gallery for a monochrome cellphone) or design elements without a semantic meaning (e.g. company logos in an online shop) can be excluded from presentations on devices with small displays or low bandwidth connections. (3) *Separation*: As an alternative to exclusion, it can be advantageous to put certain content pieces onto separate pages and automatically create hyperlinks to those pages. This mechanism is very useful to keep the structure of Web pages while providing a lot of information easily understandable on handhelds.

The mentioned examples represent static adaptation (also called *adaptability*), i.e. an adaptation that does not consider the user's browsing behavior. However, in some cases it is meaningful to consider dynamic adaptation (*adaptivity*), i.e. adaptation according to parameters that may change while the hypermedia presentation is being browsed. As possible scenarios (in presentation design) we mention the dynamic reorganization of presentation elements on a Web page when the user resizes her browser window or the delivery of alternative media instances when the available bandwidth for a mobile user is decreasing.

3 The Hera Methodology

A typical scenario in a WIS is that in response to a user query the system (semi-) automatically generates a hypermedia presentation. The generated hypermedia presentation needs to be tailored to the device characteristics and to the user preferences. Hera [1] is an example of a model-driven methodology for designing these aspects in a WIS. It proposes a sequence of steps that the WIS designer needs to follow: the conceptual design defining the data schema, the application design specifying the navigation through the data, and the presentation design describing the rendering of the data. During this entire process, possible adaptation aspects are included in the associated models [2]. This section concentrates on two of these models: the conceptual model and the application model. The presentation model (its extended definition) will be described in Section 5, after introducing the adaptive layout managers of AMACONT.

The *Conceptual Model* (CM) defines the schema of the data available in the WIS. It is composed of a hierarchy of concepts and concept relationships.

Concepts have associated attributes that refer to media items. Figure 1 gives an excerpt of the CM for our running example. Concepts are depicted in dark ovals and contain light ovals which stand for concept attributes. In the specification of relationship cardinalities the 'Set' construct is used on the side of the concept that can have more than one instance associated to one instance of the other concept.
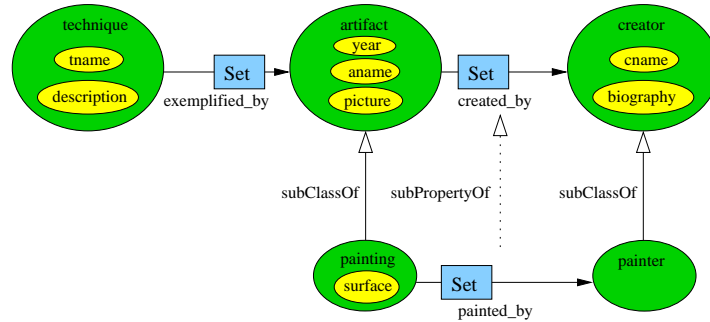

Fig. 1. The conceptual model

The media types associated to the concept attributes are described in the *Media Model* (MM), a submodel of CM. MM is a hierarchical model composed of media types. The most primitive media types are: Text, Image, Audio, and Video. Figure 2 shows an excerpt of the MM for our running example. Media types are depicted in dark rectangles.

Adaptation in the CM is based on the *conditional inclusion* of elements. Arbitrary complex conditions referencing (ranges of) data describing the device capabilities and user preferences can be used. As we do specify these conditions as XSLT conditions their expressive power is equivalent to the expressive power of XSLT conditions. This information is stored in a *user profile* containing attribute-value pairs according to a CC/PP [14] vocabulary. The light rectangles in Figure 2 depict two adaptation conditions considering screen size constraints. One condition requires that a long text for the technique description and a large image for the artifact pictures is used for PCs. The other condition stipulates that a short text for the technique description and a small image for the artifact pictures is used for PDAs. Remember that the selection of content variants with different quality is an important means of device adaptation.

The *Application Model* (AM) specifies the navigational aspects of the application. It is based on the notion of a *slice* which is a meaningful grouping of concept attributes that need to be shown together in the hypermedia presentation. A slice can be viewed as a presentation attribute associated to a concept. In this way the AM is an extension of the CM that provides a view over the CM.

The AM is composed of a hierarchy of slices and *slice relationships*. There are two kinds of slice relationships: compositional relationships (aggregation between
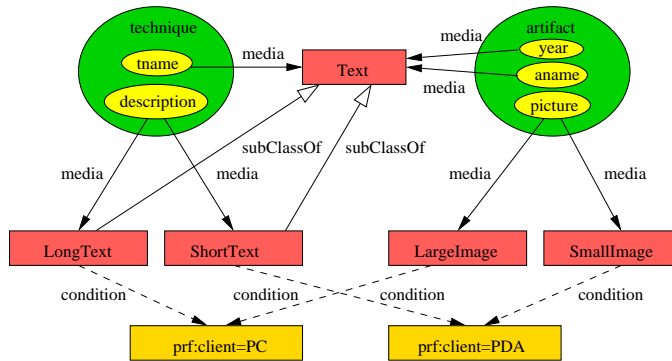
Fig. 2. The media model

slices) and navigational relationships (navigation between slices). If a compositional slice relationship relates slices associated with two different concepts, then the CM relationship involved in the composition needs to be specified. Figure 3 shows an excerpt of the AM for our running example. Slices are depicted (as their name suggests) by pizza-slice shapes. Hierarchical relationships between slices are omitted here due to lack of space (we refer the reader to [2] for details on this matter). In complete analogy to the CM, adaptation issues in the AM can be defined by attaching *appearance conditions* to slices.
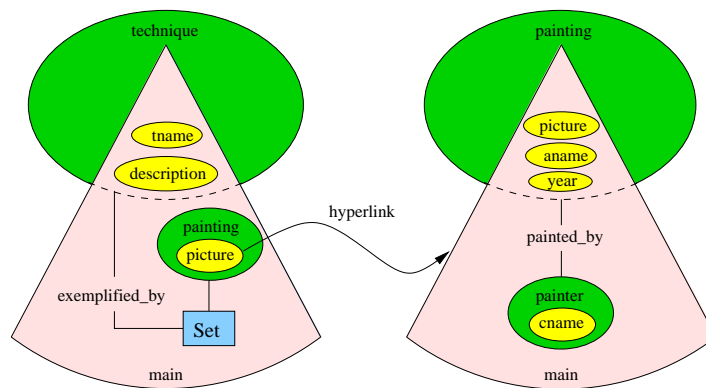

Fig. 3. The application model

4 AMACONT's Component-based Document Model

The component-based document format of AMACONT [5] aims at building personalized ubiquitous Web applications by aggregating and linking *configurable*

document components. These components are instances of an XML grammar representing adaptable content on different abstraction levels, i.e. layers in AMA-CONT (see Figure 4). *Media components* encapsulate concrete media assets by describing them with technical metadata. *Content units* group media components by declaring their layout in a device-independent way. Finally, *document components* define a hierarchy out of content units to fulfill a specific semantic role. The *hyperlink view* for defining typed links is spanned over all component layers. For a detailed introduction to the document model the reader is referred to [5].
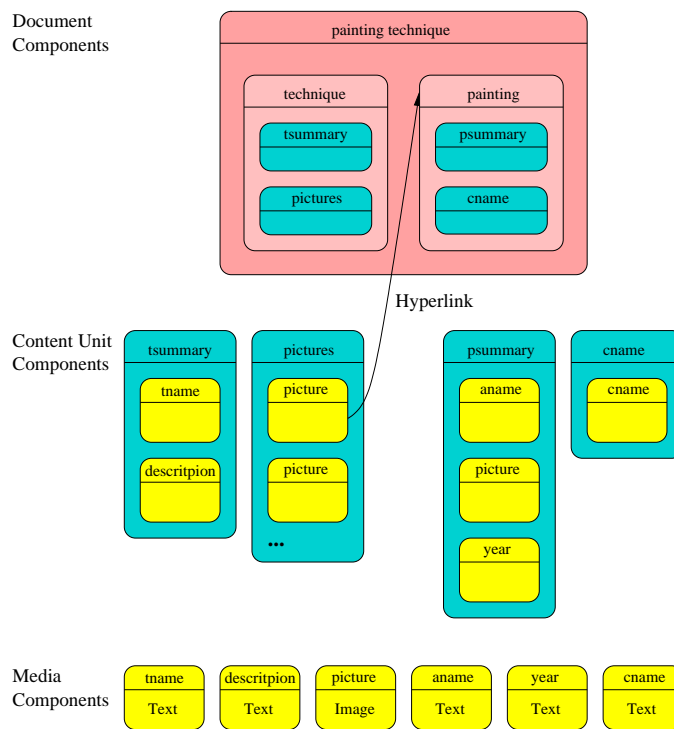
Document Components

painting technique

technique

tsummary

pictures

painting

psummary

cname

Hyperlink

Content Unit Components

tsummary

tname

descritpion

pictures

picture

picture

...

psummary

aname

picture

year

cname

cname

Media Components

tname

Text

descritpion

Text

picture

Image

aname

Text

year

Text

cname

Text

Fig. 4. The document model

In order to describe the *presentation* of component-based Web documents, AMACONT allows to attach XML-based layout descriptions [5] to components. Inspired by the layout manager mechanism of the Java language (AWT, Swing) and the abstract user interface representations of UIML or XIML [12], they describe a client-independent layout allowing to abstract from the exact resolu-

tion of the browser's display[3]. Furthermore, they also enable to declare simple presentation layer adaptation rules. Note that layout managers of a given component only describe the presentation of its immediate subcomponents which encapsulate their own layout information in a standard component-based way.

Currently four layout managers are defined. `BoxLayout` lays out multiple components either vertically or horizontally. `BorderLayout` arranges components to fit in five regions: north, south, east, west, and center. It was chosen because it strongly resembles the structure of many Web pages consisting of a header, an optional footer, a main area and one or two sidebars. `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. Though it can be realized by nested `BoxLayout`s, we implemented it separately because WISs often present dynamically retrieved sets of data in a tabular way. The following code snippet depicts the layout description of a content unit containing two media components adjusted by the layout manager `BoxLayout`. An image (aligned right) and a text object (aligned left) are arranged above each other, taking 30 and 70 percent of the available vertical space.

```
<BoxLayout axis="yAxis" border="1">
   <ComponentRef ratio="30%" halign="right">Picture1</ComponentRef>
   <ComponentRef ratio="70%" halign="left">Text1</ComponentRef>
</BoxLayout>
```

Layout managers are formalized as XML tags 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. Subcomponent attributes describe how each referenced subcomponent has to be arranged in its surrounding layout. For instance, the `halign` attribute of `Picture1` declares it to be right-justified. Table 1 summarizes the possible attributes of `BoxLayout` by describing their names, role, usage (required or optional) and possible values.

Even though most attributes are device independent, we also allowed two platform-dependent attributes in order to consider the specific card-based structure of WML presentations. 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 text of which is defined in `wml_description`. Note that this kind of content separation (see Section 2) provides scalability by fragmenting the presentation according to the very small displays of WAP-capable mobile phones.

The rendering of media objects is done at run time by XSLT stylesheets transforming components with abstract layout properties to specific output formats, such as XHTML, cHTML, and WML.

---

[3] In contrast to UIML or XIML, AMACONT offers a light-weight solution that focuses only on the rendering of components and not on other aspects like tasks, domain objects, or dialogs, that can be effectively dealt with in the CM or AM.

Layout Attributes	Meaning	Usage	Values
`axis`	Orientation of the BoxLayout	req.	xAxis|yAxis
`space`	Space between subcomponents	opt.	int
`width`	Width of the whole layout	opt.	string
`height`	Height of the whole layout	opt.	string
`border`	Width of border between subcomponents	opt.	int
Subcomponent Attributes	Meaning	Usage	Values
`halign`	Horizontal alignment of subcomponents	opt.	left|center|right
`valign`	Vertical alignment of subcomponents	opt.	top|center|bottom
`ratio`	Space taken by subcomponent	opt.	percentage
`wml_visible`	Should be shown on same WML card?	opt.	boolean
`wml_desc`	Link description for WML	opt.	string

Table 1. BoxLayout attributes

5 Putting it All Together

There are several analogies between *instances* of a Hera application model and *components* in AMACONT. Both represent meaningful presentation units bearing also some semantic role (e.g. painting or painting technique) 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 user's display and may contain adaptation issues according to device profiles and user profile parameters. However, in contrast to application model instances AMACONT components also contain adaptive layout information. Note that layout in Hera is specified in the presentation model (hence, the extensions we made to the definition of this Hera model).

Taking advantage of these analogies and the fact that both Hera and AMACONT rest upon XML, our aim was to combine the modeling power of Hera and the versatile presentation capabilities of AMACONT. Therefore, the concept of adaptive layout managers was transferred to the Hera *Presentation Model* (PM). As in AMACONT, they can be assigned to Hera slices in order to declare the arrangement of their subslices in an implementation-independent way. Furthermore, the PM schema allowing to declare such assignments was formalized in RDFS. This formalization enables the *automatic mapping* of high-level Hera AM and PM specifications to AMACONT's implementation units, i.e. components.

5.1 PM Schema

The RDFS-based Hera presentation model supports two mechanisms: the definition of layout managers and their assignment to AM slices. Based on our running example, the following code snippet from a Presentation Model demonstrates how a layout manager can be assigned to a slice. The graphical form of this PM is shown in Figure 5.

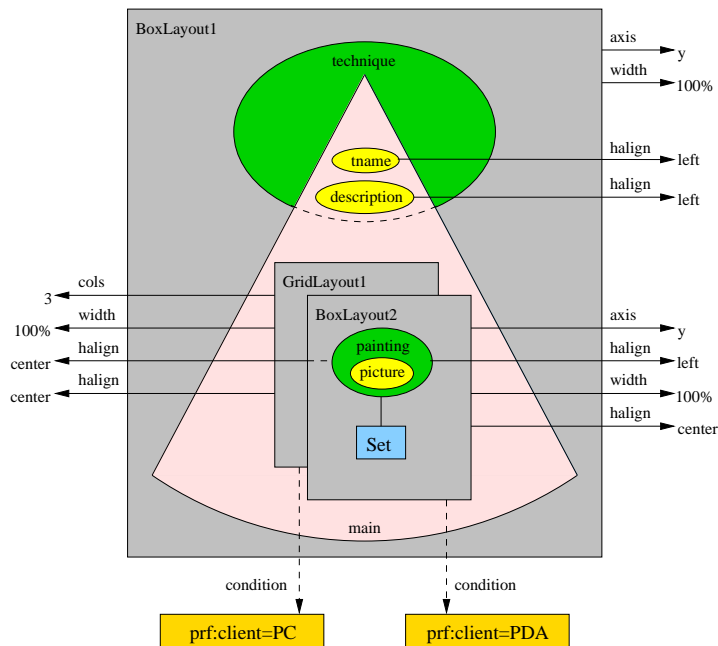**Fig. 5.** A PM example

```
<Slice rdf:about="#Slice.technique.main">
     <layout rdf:resource="#BoxLayout1"/>
</Slice>
```

This assignment rule implies to visualize the slice `Slice.technique.main` according to the layout manager `BoxLayout1`. The following example shows how `BoxLayout1` is specified in detail.

```
<BoxLayout rdf:id="BoxLayout1">
   <axis>y</axis>
   <width>100%</width>
   <slice-ref rdf:resource="Slice.technique.tname"
              pres:halign="left"/>
   <slice-ref rdf:resource="Slice.technique.description"
              pres:halign="left"/>
   <access-element-ref rdf:resource="SetOfLinks_1"
              pres:halign="center"/>
</BoxLayout>
```

Attributes for components of AMACONT's layout manager (Section 4) have been adopted to describe the spatial adjustment of subslices. Both attributes describing the overall layout and attributes specifying the arrangement of each referenced subslice (`slice-ref`) or access element (`access-element-ref`) can be defined. Still, the layout manager of a slice only specifies how its immediate

subslices are to be rendered. For the access element `SetOfLinks_1` the layout specification might look like this:

```
<Access-element rdf:about="#SetOfLinks_1">
 <layout rdf:resource="#GridLayout1"
         pres:condition="pres:client='PC'"/>
 <layout rdf:resource="#BoxLayout2"
         pres:condition="pres:client='PDA'"/>
</Access-element>
```

Note the attribute `pres:condition` that allows to declare simple *adaptation conditions* that reference parameters from the user/device profile. Whereas for example the paintings exemplifying the presented painting technique are arranged on a PC in a tabular way (GridLayout1), the small screen size of PDAs requires adjusting them below each other (BoxLayout2).

In contrast to AMACONT's layout managers (defined at instance level), Hera layout assignments have to be specified at *schema level*. Due to the dynamic nature of WIS applications, this means that the number of items in an access element is not known at design time. In such cases one should use either a `BoxLayout` with an undefined number of cells or a `GridLayout` so that only one of its dimensions (`columns` or `rows`) is predeclared. The missing dimensions are automatically computed at run time.

5.2 The Data Transformation Process

Figure 6 gives an overview of the data transformation process. It is composed of four transformation steps that are described in detail in the rest of this section.

In the first step, based on the attribute values from the user/platform profile, the conditions included in the different models are evaluated. Elements having a condition evaluated to false are removed from their corresponding models. As a result, an adapted conceptual model, an adapted application model, and an adapted presentation model are obtained. Note that this adaptation is done at the schema level. The rest of the transformation steps will be performed at the instance level (or data level).

The input data to the transformation pipeline is the conceptual model instance. The gathering of this data in response to a user query is outside the scope of this paper, see [1]. In the second step the conceptual model instance is used to populate the application model. The resulting application model instance represents the presentation navigational structure.

The third step aims at *mapping* AM instances to adaptable AMACONT components, and is realized by two transformations substeps. In the first substep slice composition references are resolved (*unfolding*) and AM instances with hierarchical slice structures (slices containing subslices) are created. In the second substep the unfolded AM instances are automatically *transformed* to hierarchical AMACONT component structures. First, top-level slices are mapped to top-level document components. Then, subslices and their attributes are recursively mapped to subcomponents according to the following rules:
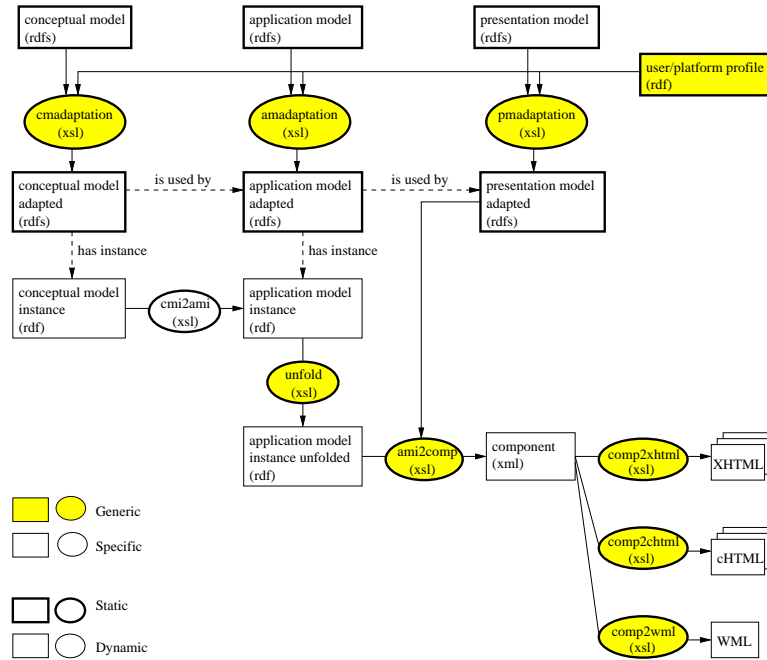
Fig. 6. The data transformation process

1. Concept attributes are mapped to media components. `Integer` and `String` attributes are assigned to text components, media attributes to corresponding media components (image, audio, video etc.).
2. Slices containing concept attributes from a single concept are 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 are mapped to composite document components containing child document components for each aggregated subslice. For those subslices this mapping process has to be performed recursively.

The *layout attributes* of the created AMACONT components are *configured* according to the PM schema. Beginning at top-level document components and visiting their subcomponents recursively, the appropriate AMACONT layout descriptors are added to the meta-information section of each component's header. Since the layout manager attributes of the Hera PM rest upon the layout concepts of AMACONT, this mapping is a straightforward process. Yet, for access elements (containing a variable number of subslices depending on a dynamic query) the concrete dimensions of `BoxLayout` or `GridLayout` have to be recalculated for each particular user request.

In the last (fourth) step the components are transformed to the corresponding output format. For instance, a `BoxLayout` in XHTML is realized by means of a

table (and its specific attributes) with either one column or one row. However, not all layout managers can be visualized properly on all devices. As an example, since PDAs or WAP phones have very small displays, a horizontal `BoxLayout` is automatically converted to a vertical arrangement of subcomponents on those devices. Similarly, in the case of `OverlayLayout` only the upper component is presented on handhelds. Figure 7 shows two possible outputs of our running example, one for a PC (XHTML) and another for a PDA (cHTML).
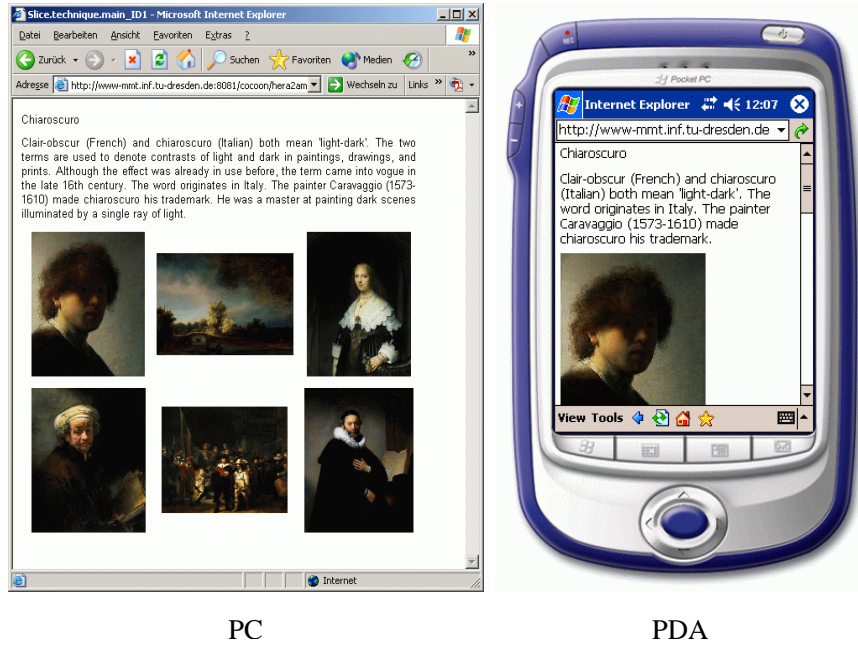

PC   PDA

Fig. 7. Desktop/PDA presentation

6   Conclusion and Future Work

In this work we have illustrated the role of presentation layer adaptation in WIS design. We have demonstrated how the Hera specification framework (and its implementation tools) can be complemented with the flexible implementation layer provided by the AMACONT project. Thus these high-level specifications can be mapped to components to be published in different output formats. It has resulted in an integrated framework (and tool set) that combines the capabilities of Hera at the conceptual and navigation levels with those of AMACONT at the presentation level.

A major focus in ongoing work concentrates on the dynamic coupling of Hera and AMACONT. The static presentation involved in adaptable Web presenta-

tions has to be enhanced by feedback mechanisms that allow to dynamically react to user's browsing behavior. For this, interactions have to be captured in the generated presentation and sent to the application layer, so that new application model instances can be created and passed back to the presentation layer (engine). With this extension we will also be able to deal with dynamic adaptation (adaptivity) in the joint system.

References

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

[2] Frasincar, F., Barna, P., Houben, G.J., Fiala, Z.: Adaptation and reuse in designing web information systems. In: International Conference on Information Technology, Track on Modern Web and Grid Systems, IEEE Computer Society (2004) 387–391

[3] Lassila, O., Swick, R.: Resource description framework (rdf) model and syntax specification. W3C Working Draft. (22 February 1999)

[4] Brickley, D., Guha, R.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft. (10 October 2003)

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

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

[7] Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann (2003)

[8] Koch, N., Kraus, A., Hennicker, R.: The authoring process of the uml-based web engineering approach. In: First International Workshop on Web-Oriented Software Technology. (2001)

[9] Pastor, O., Fons, J., Pelechano, V.: Oows: A method to develop web applications from web-oriented conceptual models. In: International Workshop on Web Oriented Software Technology (IWWOST). (2003) 65–70

[10] Gomez, J., Cachero, C. In: OO-H Method: extending UML to model web interfaces. Idea Group Publishing (2003) 144–173

[11] van Ossenbruggen, J., Geurts, J., Cornelissen, F., Hardman, L., Rutledge, L.: Towards second and third generation web-based multimedia. In: WWW10, The Tenth International Conference on the World Wide Web, Hong Kong, ACM (2001) 479–488

[12] Souchon, N., Vanderdonckt, J.: A review of xml-compliant user interface description languages. In: Conference on Design, Specification, and Verification of Interactive Systems. Volume 2844 of Lecture Notes in Computer Science., Springer (2003) 377–391

[13] Eisenstein, J., Vanderdonckt, J., Puerta, A.: Applying model-based techniques to the development of uis for mobile computers. In: 6th International Conference on Intelligent User Interfaces, ACM (2001) 69–76

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