Journal of Web Engineering, Vol. 0, No. 0 (2003) 000–000
© Rinton Press

Engineering Semantic Web Information Systems in Hera

RICHARD VDOVJAK, FLAVIUS FRASINCAR, GEERT-JAN HOUBEN, and PETER BARNA
Department of Computer Science, Eindhoven University of Technology, PO Box 513
Eindhoven, NL - 5600 MB, the Netherlands
{*richardv, flaviusf, houben, pbarna*}*@win.tue.nl*

Received (received date)
Revised (revised date)

The success of the World Wide Web has caused the concept of information system to change. Web Information Systems (WIS) use from the Web its paradigm and technologies in order to retrieve information from sources on the Web, and to present the information in terms of a Web or hypermedia presentation. Hera is a methodology that supports the design and engineering of WIS. It is a model-driven methodology that distinguishes three parts in the design: integration, data retrieval, and presentation generation. The integration part manages the gathering of data from different sources on the basis of source ontologies and mappings between those source ontologies and the conceptual model of the WIS. The data retrieval part handles the user queries and produces the data that represents the query result. In the presentation generation part this query result is transformed into a Web presentation and that presentation is constructed to suit the user (platform), e.g. HTML, WML, or SMIL. In this paper we address the Hera design methodology and specifically explain two models: the integration model that covers the different aspects of integration, and the adaptation model that specifies how the generated presentations can be adaptable (e.g. based on device capabilities, user preferences) and adaptive (e.g. based on user browsing history). This detailed description includes an explanation of how the Hera software is constructed. This software provides a set of transformations that allow a WIS to go from integration to presentation generation. These transformations are based on RDF(S), the foundation of the Semantic Web. We show how RDF(S) has proven its value in combining all relevant aspects of WIS design, thus illustrating how Hera allows the engineering of Semantic Web Information Systems (SWIS).

Keywords: WIS, SWIS, Hypermedia, RDF(S), Semantic Web, XSLT

Communicated by: to be filled by the Editorial

1 Introduction

It is fair to say that the World Wide Web is the most popular source of information. The number of its users and the attention it attracts speak for themselves. The overwhelming success of the Web and its considerable influence on the way in which we exchange information, cause some people to compare it to Gutenberg's invention of the printing press. It is easy to observe how computer applications make information available for a very diverse audience on different platforms worldwide and 24 hours a day.

In the spirit of the Web, the nature of information systems as we know them has changed. The early applications on the Web presented data in terms of carefully authored hyper(media) documents. Typically, the author hand-crafted a static collection of pages and links between these pages in order to convey information to the users. At the time this use of hypermedia was already a step forward and contributed to the popularity of the Web. When more and more (existing) data sources got connected to the Web, more information became available and we saw a trend for the typical Web application towards a data-intensive application. The application uses for example data generated from structured files or databases, usually in an on-the-fly manner, in order to deliver the right information to the user. At the same time this trend has significantly influenced and changed the existing notion of information system: a modern professional information system has become a Web application. In that light we consider Web Information System (WIS) [1] as an information system that uses the Web paradigm (and technologies) to retrieve information from the sources and deliver it to the users.

It is typical for the Web that a WIS needs to bridge the gap between a collection of heterogeneous and dynamic data sources and a group of users with different preferences using different platforms for accessing the information. This aspect of WIS makes the early proposals for designing and implementing Web or hypermedia applications not applicable anymore [2]. Developing a hyperdocument typically meant a mix of content and presentation design, but also a lot of ad-hoc programming. The data-intensive and dynamic nature of a WIS requires a more rigorous development process. One reason is that the one-size-fits-all approach that is so typical for traditional hypermedia is not suitable for delivering information at run-time to different users with different platforms (e.g. PC, PDA, WAP phone, WebTV) and different network connections (e.g. dial-up modem, network copper cable, network fiber optic cable).

A characteristic aspect of WIS is the semi-automatic generation of a hypermedia presentation for the data that the application delivers to the user. Due to the dynamical nature of the data in the application the hypermedia output needs to be generated automatically by the application, and the main issue in the application design is therefore the specification of this hypermedia generation process. True Web engineering approaches offer designers and programmers a design framework often based on a model-driven approach, specifying the different aspects of the complete application design in terms of separate models, e.g. for data (content) and hypermedia presentation.

Personalization is typical for the hypermedia design in a WIS, taking into account the requirements from the users, e.g. user preferences, and the requirements from their platforms, e.g. device capabilities and network connection. This personalization is one example of adaptation [3] in the design process. Adaptation tries to overcome the problems of the one-size-fits-all approach in the traditional hypermedia. It can do so by generating a hypermedia presentation that suits the context of the browsing: we say that the presentation is adaptable. Adaptation can also be based on the user's history of browsing in the presentation: in that case we say that the presentation is made adaptive.

Integration is another issue that is characteristic for WIS [4]. Usually, the content of the system is gathered from different sources that are distributed over the Web. This requires the design of the WIS to include specifications of which sources to select and how to map the data in the sources to the data (model) in the WIS (both in terms of schema integration and data integration). With such specifications the WIS can retrieve the data that needs to be included in the presentations that are generated. The approaches and techniques used for integration are inspired by similar approaches for the traditional integration of databases: their application in the context of WIS appears useful for WIS design.

Several Web engineering frameworks already exist, e.g. Araneus [5], Strudel [6], WebML [7], UWE [8], XAHM [9], OntoWebber [10], and XWMF [11]. Our Hera methodology [12] targets Web engineering, distinguishes models for the conceptual content and for the hypermedia aspects of the application, and makes integration and adaptation central issues in its models. The associated Hera (software) framework includes a number of steps to get from data retrieval to presentation generation at the level of instances. These steps basically build a sequence of data transformations that in the end present the right data in the right format (e.g. HTML, WML, or SMIL) to the user. Knowing the advantage of Web application interoperability we have chosen to use Semantic Web technology, specifically RDF(S) [13, 14], in the experimental prototype that illustrates Hera in this paper: we use RDF(S) to express the data used in different steps of the transformation process. With the aid of XSLT [15] we transform the RDF data in order to (in the end) generate presentations in HTML, WML, or SMIL.

This paper shows the model-driven approach of Hera, described earlier in [12], and introduces the transformation software that builds the heart of the hypermedia presentation generation process. These transformations help to generate step-by-step the hypermedia presentation for the data requested by the user. Having chosen RDF(S) as the primary format for the data to be transformed, a nice advantage is that it is easy to re-use parts of the data transformation process in connection with other (outside) components. This allows the different Hera's software modules to be combined with external software modules and/or data. Moreover, the paper specifies how ontologies expressed in RDF(S) are crucial aids in the entire process of integrating and retrieving data and generating presentations. With this central role for RDF(S), the foundation of the Semantic Web, we demonstrate how Hera supports the engineering of Semantic Web Information Systems (SWIS).

2 Hera Methodology

A primary focus of the Hera project [12] is to support WIS design and implementation, particularly the hypermedia aspects. A WIS generates a hypermedia presentation for data that in response to a user query is retrieved from the data storage. This entire process of retrieving data and presenting it in hypermedia format needs to be specified during the design of the WIS, since the WIS has to be programmed in such a way that it can automatically execute that process.

2.1 WIS Architecture

The typical structure of the WIS architecture in the Hera perspective is given in Figure 1 in terms of three layers:

- The Semantic Layer defines the content that is managed in the WIS in terms of a conceptual model. This layer includes the definition of the process of integration needed to gather the data from different sources. If the data is made available from outside the WIS, a search agent or information retrieval engine could be the interface to the WIS.

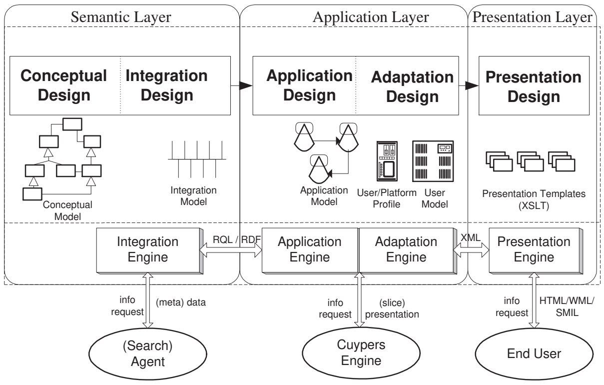

Fig. 1. The Hera methodology/suite.

- The Application Layer defines the abstract hypermedia (navigation) view on the data in terms of an application model, which represents the structure shown to the user in the hypermedia presentation. This layer includes the definition of the adaptation in the hypermedia generation, e.g. based on a user model and a user/platform profile. If desired, the Application Layer can interface not with the own Presentation Layer, but with an "outside" Presentation Layer like the Cuypers engine [16].

- The Presentation Layer defines the presentation details that together with the definitions from the Application Layer are needed for the generation of a presentation for a concrete presentation platform, e.g. HTML, WML, or SMIL. It outputs this presentation to the user's browsing platform.

2.2 Methodology Phases

Hera's approach to build the application model on top of the conceptual model has a significant advantage: it facilitates model-driven transformations to populate the model of the application with the retrieved data. In fact, the Hera methodology that aligns with the architecture from Figure 1 distinguishes the different data transformations that are necessary to generate the hypermedia output in response to a user query. For the sake of clarity we distinguish the following phases:

- integration and data retrieval

- presentation generation

The first combined phase of integration and data retrieval helps to make the data available from different sources, such that in response to a *user query* a *conceptual model instance* is generated that contains the data for which the application is going to generate a presentation: see Figure 2[a].

[a]In these figures the ellipses denote the transformations (in XSLT or Java), and the squares denote models or data. The shapes in grey denote application-independent items, the shapes in white with bold lines denote application-dependent items, while the others are query-dependent items.

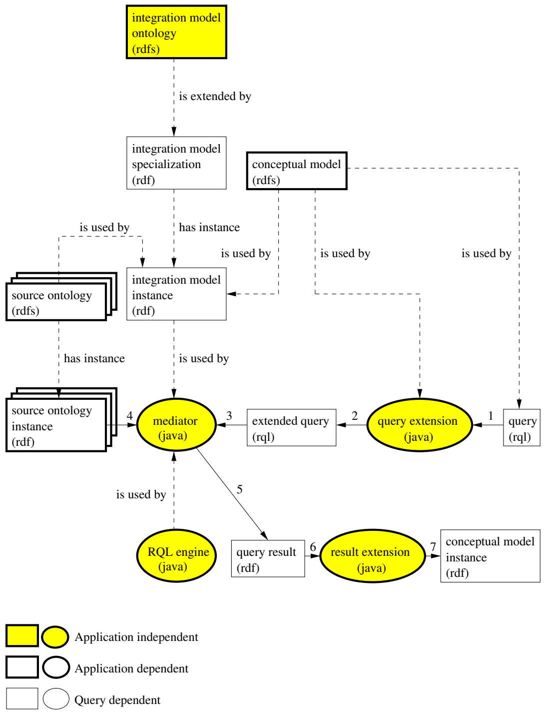

Fig. 2. Integration and data retrieval.

The integration is in principle performed before querying, as opposed to the retrieval and presentation generation that are performed for every query. It represents the data stored and therefore uses an integration model to map the data from the different sources into concepts of the conceptual model. From a mapping at schema (ontology) level a mapping at instance level is derived. This mapping is needed whenever for a given query, the instances that compose the query result need to be retrieved. These instances need to be extracted by the mediator from the different *source ontology instances*. The role of the integration is to make (on-demand) these source ontology instances available.

The data retrieval handles the reception of the *user query*, and in response produces a *conceptual model instance* for the query result. It starts with the translation of the query formulated by the user into a query that can act as a retrieval request on the data stored. This translation takes into account that while the user is allowed to formulate a query by mentioning items from the conceptual model or application model, the application model defines exactly which concepts need to be retrieved in connection with the items mentioned: this is known as query extension. Subsequently, using the query engine, the mediator retrieves the data from the sources and provides the query result. Finally, this query result needs to be transformed into the conceptual model instance that is passed on to the phase of presentation generation.

In the presentation generation phase, for the retrieved data a hypermedia presentation is generated: see Figure 3. It starts from the data that builds the result of the user query, represented by the *conceptual model instance*. This data is transformed in three main steps

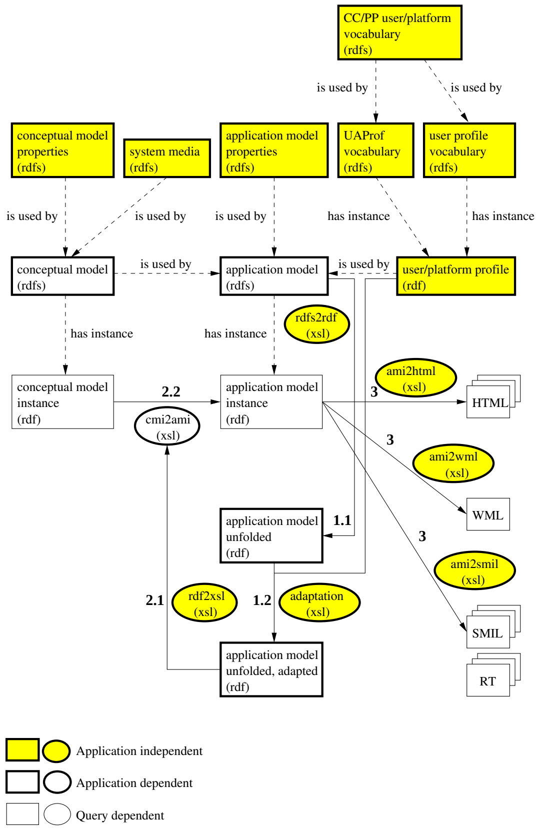

Fig. 3. Presentation generation.

into a presentation suited for the user's platform. In the last step the presentation gets generated for the specific platform. Before that, the data from the conceptual model instance is transformed into the *application model instance* that adheres to the application model. Before this transformation can be applied, the transformation specification has to be generated first based on application model and considering adaptation. The adapted application model instance is serialized in the specific format of the user's browser (e.g. HTML, WML, or SMIL).

We want to remark that in our current approach we primarily look at hypermedia applications. This implies that the main interaction from the user is based on navigation: following links. In full-fledged Web applications other kinds of user input can exist, e.g. entering values in forms. In on-going research we are considering how other kinds of user interaction can be incorporated in the design, but for the methodology as we describe it in this paper we restrict ourselves to the interaction through navigation.

2.3 RDF(S)

RDF(S) is the main format used in our data transformations. One of the reasons for choosing RDF(S) is that it is a flexible (supporting schema refinement and description enrichment) and extensible (allowing the definition of new resources/properties) framework that enables Web application interoperability. An example of application interoperability is the usage of different navigation ontologies for a given application domain ontology. In Hera, model instances are represented in plain RDF validated against their associated models (schemas) represented in RDFS.

The use of RDFS allows us also to reuse existing RDFS vocabularies like the User Agent Profile (UAProf) [17], a Composite Capability/Preference Profiles (CC/PP) [18] vocabulary for modeling device capabilities and user preferences [19]. With RDF(S) we also have an effective format to deal with the semi-structured nature of the data involved.

2.4 Tools

While RDF(S) seems suitable for the specification of the different Hera models, a disadvantage is the lack of full-fledged RDF(-aware) transformation processors. Since the Hera models and their instances can be treated as plain XML representations, we have an XSLT processor perform the different transformations based on stylesheets. For the purpose of presentation generation this approach is sufficient as we do not use RDF(S) inference rules (e.g. transitivity of inheritance) in the transformation specification (those are resolved during the data retrieval phase). In our first prototype we used Xalan 1.2D02 as the XSLT processor (XPath 1.0, XSLT 1.0). Since our demands include multiple outputs for one stylesheet and procedural constructs like variable assignment and loop operators, we replaced Xalan in the presentation generation phase with the more powerful Saxon 7.0 (XPath 2.0, XSLT 2.0). For the purpose of data retrieval we use the most advanced RDF(S) query language to date, RQL [20] and its Java-based interpreter called Sesame [21]. This combination proved to be useful when building our retrieval engine, which in fact acts as a distributed RQL query engine.

2.5 Related Work

Most of the Web engineering approaches do not consider integration and adaptation, as opposed to what is the case in our Hera methodology.

We mention Strudel [6] as a related approach consisting of a data model, a query language and an HTML template language. In comparison to Hera it offers a more mature implementation in terms of querying data. However, we observe that it is based primarily on the structure of the data. Before coming to the HTML templates that data structure, represented in the data model, acts as the main view on the data. For the same goal Hera separates clearly the views for storing data and for accessing data (in Conceptual Model and Application Model). This separation allows to reason about the hypermedia aspects independently from the data storage: the design of the navigation and access is thus effectively separated from the structure. In order to generate hypermedia presentations this layered approach appears useful.

OntoWebber [10] is another model driven approach to managing data intensive Web sites. Similar to Hera it covers also the integration part with the difference that the data is materialized in a central repository whereas Hera retrieves the data on-demand. OntoWebber also supports adaptability in the form of personalization; Hera goes one step further offering adaptivity based on the user model and the adaptation model.

Concentrating on adaptation we note that models like AHAM [22] or MRM [23] (that is connected to UWE) are reference models that specifically consider adaptation. However, these reference models do so primarily in the context of adaptive hypermedia documents. The generation of hypermedia presentations in WIS poses different requirements: the presentation of data elements from the dynamic contents of a WIS is essentially different from the presentation of a document (structure).

Next to these reference models for adaptation we see only a few of the concrete Web engineering approaches that deal with adaptation. As notable exceptions among those approaches we mention XAHM [9], an XML-based methodology, UWE [8], a UML-based methodology, and XWMF, an RDF-based modeling framework. Given its RDF-based nature we address XWMF here in more detail.

The eXtensible Web Modeling Framework (XWMF) [11] consists of an extensible set of RDF schemas and descriptions to model Web applications. The core of the framework is the Web Object Composition Model (WOCM), a formal object-oriented language used to define the structure and content of a Web application. WOCM is a directed acyclic graph with complexons as nodes and simplexons as leaves. Complexons define the application's structure while simplexons define the application's content. Simplexons are refined using the subclassing mechanism in different variants corresponding to different implementation platforms. While Hera provides both a modeling framework and a methodology for developing Web applications, XWMF appears to be only a modeling framework.

3 Integration and Data Retrieval

The main task of the integration and data retrieval phase is to connect the conceptual model with several autonomous sources by creating channels through which the data will populate on request the concepts from the conceptual model. This involves identifying the right concepts occurring in the source ontologies and relating them to their counterparts in the conceptual model. Note that as opposed to classical database schema integration we do not aim at integrating all source concepts, but rather select only those that are relevant with respect to the defined conceptual model.

3.1 Conceptual Model

The conceptual model (CM) provides a uniform semantic view over multiple data sources. The CM serves as an interface between data retrieval and presentation generation. The CM is composed of concepts and concept properties that together define the domain ontology. There are two types of concept properties: concept attributes which associate media items to the concepts and concept relationships that define associations between concepts.

As depicted in Figure 3 a CM is expressed in RDFS using two additional RDFS descriptions: CM properties and system media types. The CM properties define the `cardinality` and `inverse` of concept relationships. The system media types define the multimedia ontology. The root of this ontology is the `Media` class which is further refined in `Text` and `Image` classes. `Text` is characterized by the `length` attribute (expressed in number of characters) and `Image` has `width` and `height` attributes (expressed in pixels) plus a string containing a URL of the actual picture. `Text` has two subclasses `Integer` and `String`. Other media types can be easily added to the present multimedia ontology.

The running example used throughout the paper describes the design of a WIS serving as a virtual art gallery that allows visitors to create on-the-fly exhibitions (browseable presentations) featuring their favorite painters, paintings, and painting techniques. These are assembled on demand, based on the visitor's query, from the exhibits coming out of different

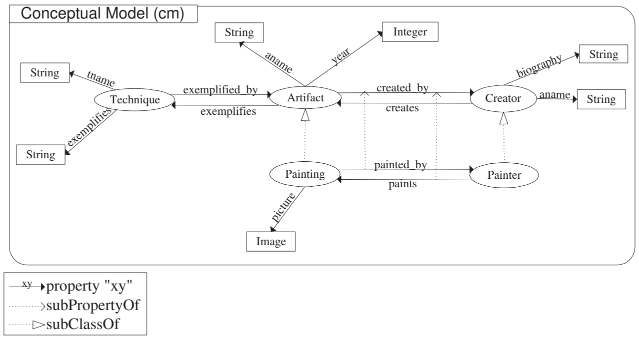

Fig. 4. Conceptual model.

(online) museums and annotated with relevant descriptions from an online art encyclopedia. All this data is offered from a single entry point, semantically represented by the CM.

Figure 4 presents the CM of our example. It defines a domain ontology composed of five concepts and three concept relationships together with their inverse counterparts. Each concept has specific concept attributes associated. Note that the graphical syntax used in Figure 4 resembles the graphical syntax of RDF(S), extended with the media types and special edges representing built in RDFS properties like `subClassOf` and `subPropertyOf`. Other (application) properties are denoted as full-arrow edges and are attached directly to classes (denoted as ovals) that represent their domains and ranges.

3.2 Sources

In previous research [4, 24] we discussed how to overcome the syntactic heterogeneity of different source formats by introducing a layered approach starting with a layer of wrappers. In this paper we focus on the issues regarding the semantic heterogeneity. We consider for integration only those sources that are capable of exporting their schema in an RDFS based ontology and their data upon request (an RQL query) in RDF. In other words, we assume that each source offers its data on the Semantic Web platform providing RQL query services.

It is often the case on the Web that the information is duplicated and offered (possibly with a different flavor) from several sources. We group such sources into semantically close clusters and provide a means to order them dynamically within a cluster, based on several notions of quality introduced by the designer. Sources within a cluster do not necessarily have the same structure but should provide approximately the same semantic content. Sometimes for the sake of simplicity we abuse the word source when in fact we mean a whole cluster represented by that source.

In our example we use two such sources/clusters to fill the CM with data. The source ontologies describing schemas of these sources are presented in Figure 5. The first source is an online encyclopedia providing data about different art pieces, offering their title, date of creation, author, used technique etc. This source yet rich in content is purely text-based. So if we want to obtain an actual image of a painting we have to consult the second source. This source represents an online multimedia catalogue of exhibits of different kinds including their digitalized versions. The literal value of the property `visualized` holds a URL of an image file that depicts the exhibit (painting). Note the abbreviated names in parentheses, which

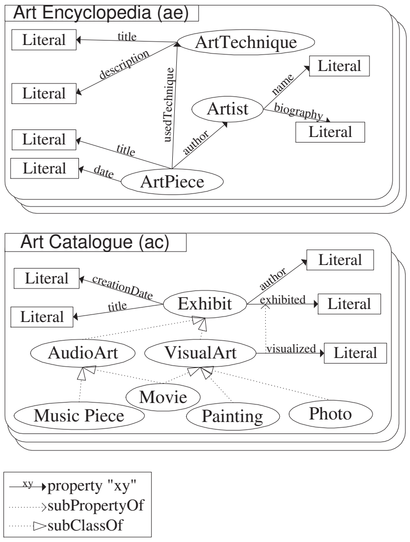

Fig. 5. Integrated sources.

denote different namespaces that later will uniquely identify the sources in the integration model[b].

3.3 Integration Model

The integration model (IM) addresses the problem of relating concepts from the source ontologies to those from the CM. This problem can also be seen as the problem of merging or aligning ontologies. The approaches to automate the solution to this problem are usually based on lexical matches, relying mostly on dictionaries to determine synonyms and hyponyms; this is however often not enough to yield good results. Even when the structure of ontologies is taken into account the results are often not satisfactory especially in the case of uncoordinated development of ontologies across the Web [25]. For this reason and for the fact that every mistake in the integration phase will propagate and get magnified in all the subsequent phases, we currently rely on the designer or a domain expert to articulate CM concepts in the semantic language of sources. What we offer the designer, is an integration ontology by instantiating which he specifies the links between the CM and the sources.

3.3.1 Integration Model Ontology

The integration model ontology (IMO) depicted in Figure 6 is a meta-ontology[c] describing integration primitives that are used both for ranking the sources within a cluster and for specifying links between them and the CM. The IMO is expressed in RDFS allowing the designer to tailor it for a particular application. The main concepts in the IMO are `Decoration` and `Articulation`.

[b] The source namespace is unique also within a cluster.
[c] IMO is a meta-ontology in the sense that its instances are dealing with concepts from other ontologies (source ontologies and the CM).

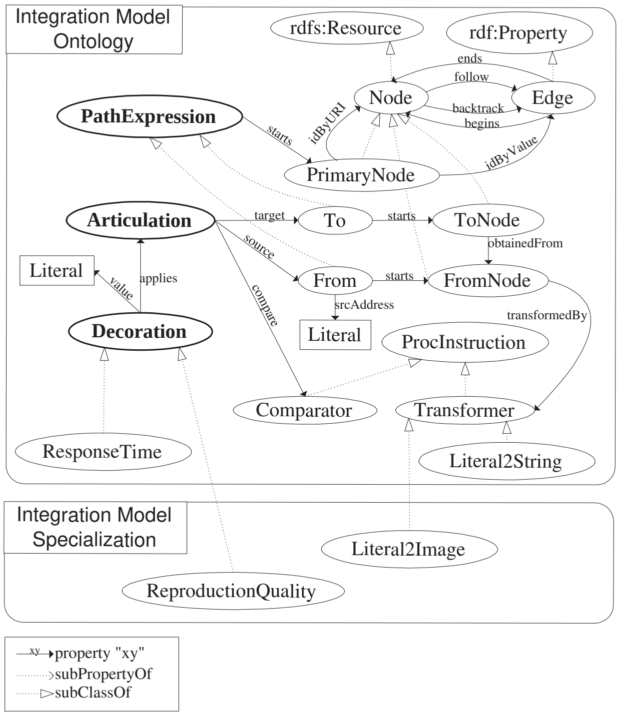

Fig. 6. Integration model ontology and its specialization.

3.3.2 Decorations

Decorations serve as a means to label "appropriateness" of different sources (and their concepts) grouped within one semantically close cluster. By having a literal property `value` they offer a simple way of ranking otherwise equivalent sources from several different points of view. There are some general decoration classes that are predefined in the framework (e.g. `ResponseTime`). However, which ordering criteria are of interest depends mostly on the application. That is why the concept of `Decoration` is meant to be extended (specialized) by the designer. In this way we allow the designer to capture in the IM his (mostly background) knowledge regarding the sources. For instance in our painting gallery example sources in the catalogue cluster are graded based on the reproduction quality of the digitalized paintings they offer. Hence the `ReproductionQuality` decoration is introduced as shown in Figure 6 bottom.

The idea behind decorations is to capture "reputation" of sources in different areas. By making this explicit, the mediator which is responsible for evaluation of queries, can consult the relevant sources in the optimal way w.r.t. the chosen ordering criterion.

3.3.3 Articulations

Articulations describe actual links between the CM and the source ontologies and clarify also the notion of the concept's uniqueness which is necessary to perform joins from several sources.

Before we explain the concept of `Articulation` (see Figure 6) we need to introduce the notion of a path expression. A path expression is a chain of concepts (represented by the class `Node`) connected by their properties (represented by the class `Edge`). If the property has the given node as its domain (in other words we follow the arrow in the RDF graph) we connect them with the `follow` meta-property. If the property has the given node as its range (going against the arrow in the graph), we connect the two by the `backtrack` meta-property. This allows us to define inverse relationships even in the case when they are not present in the source ontologies.

Each path expression starts with a link to `PrimaryNode` which is a special node that can be uniquely identified either by a URI (`idByURI`) or by value (`idByValue`). The first points to a resource whose URI serves as an ID, the second points to a property (the `Edge` type) the value[d] of which serves as an ID.

In our example we chose to use identification-by-value due to the simpler way of deciding whether two resources describe the same thing. In the case of URI identification the decision whether two URIs refer to the same real-life object would require some sort of a Web institution that normalizes several URIs to a canonical URI. This is in our opinion rather restrictive and a value-based identification is likely to work better at least in the domains that we present in our example.

An articulation contains two path expressions: the target path expression `To` pointing into the CM and the path expression called `From` pointing to a source (note the `srcAddress` property, value of which is the source URL). The target path expression contains nodes of type `ToNode`[e] that extend the `Node` with two properties: `obtainedFrom` and `transformedBy`. The first links this node to its counterpart in the `From` path expression, the second points to a converting processing instruction called `Transformer`, which is called by the mediator to transform the source to the target. Processing instructions are resources containing a piece of Java code, an XSLT transformation, an RQL query or a combination of those. They are used by the mediator for changing and comparing values. Some general processing instructions are provided by the framework (e.g. the `Literal2String` transformer); those that are application-dependent are introduced in the specialization of IMO by the designer (e.g. the `Literal2Image` transformer).

3.4 Integration Model Instance

The integration model instance is produced by the designer by instantiating the IMO[f]. Even though it is an ontology instance, it deals with the sources and the CM at the schema level, i.e. it makes statements about their concepts, not about instances. The choice of using RDF(S) as our underlying format proved to be very useful since it is easy in its distributed fashion to make statements about other resources. Figure 7 shows three articulation examples.

The first is a simple articulation linking the `cm:Technique` and its property `cm:tname` with their counterparts from the art encyclopedia source. The prime nodes are defined by the value of the labeling properties: `cm:tname` and `ae:title`.

The second articulation defines `cm:exemplified_by`. There is no direct counterpart for this relationship in the art encyclopedia, however there exists its reverse `ae:usedTechnique`.

[d]By a value of a property we mean the object in RDF terminology.
[e]The ToEdge property is defined in a similar way but was omitted in order to simplify the figure.
[f]Currently we are involved in building tools that allow the designer to specify the links and generate the articulations in the RDF(S) format automatically.

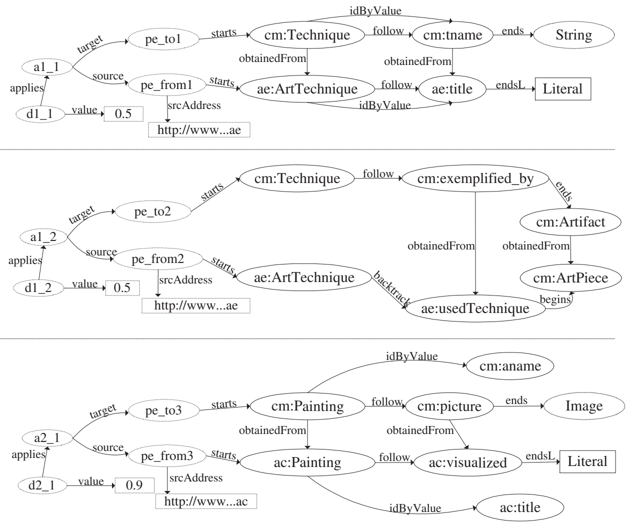

Fig. 7. Path expressions in the integration model instance.

This relationship is reached in the `from` path by means of the aforementioned `backtrack` meta-property[g].

The third articulation maps the concept `cm:Painting` and its property `cm:picture` with their counterpart from the art catalogue. It also defines that each painting is uniquely identified by the value of the property `cm:aname`.

Note that these articulations were simplified in the sense that the links to the processing instructions (i.e. transformers that transform source values into the target values) are omitted.

3.5 Data Retrieval

While the integration phase (instantiating the IM) is performed only once, prior to the user asking the query, the data retrieval phase is performed for every query. In this phase the query is extended and split into several sub-queries which are then routed to the appropriate sources. Subsequently the results are gathered and transformed into a CM instance. Figure 2 shows the dataflow of this phase with three processing blocks involved: the query extension, the mediator, and the result extension.

3.5.1 Query Extension

The query language used in our system is RQL [20], more precisely the sub-part of RQL consisting of `Select-From-Where` clauses that the user can generate by the EROS user interface [26]. During the query extension sub-phase the user query, an example of which is depicted in Figure 8, is extended to contain all relevant data which is used by the presentation generation phase.

The extension algorithm traverses the CM from a given concept(s) (`X:Technique`) and adds all concepts and/or literal types that can be reached by following property edges in the CM graph. The algorithm stops the traversal in a certain direction after it reaches a literal

[g]The use of this mechanism in fact evokes a query that determines the appropriate instances.

node or there is no property to follow from the currently examined resource. The algorithm disregards the RDF-system resources (`rdfs:Resource`, `rdf:Property` etc.).

The extended query usually covers a reasonably larger part of the CM than the original query. This is in accordance with the fact that the aim of this phase is to "prefetch" some data to enable the presentation generation engine to produce a full-fledged hypermedia presentation (not just a list of top results). In some cases (depending on the structure of the CM and the original query), this approach may however return a result which is too extensive, i.e. it is not likely that the user will browse the whole generated presentation. To avoid unnecessary retrieval the system can be parameterized by the maximal traversal depth. An alternative approach is to use a heuristic that assigns priorities to concepts from the CM and those are then considered for the (final) extended query only if their priority reaches a set threshold. Figure 9 depicts the result of the query extension for the mentioned user query.

```
select X
from {X:Technique}tname{Xtname}
where Xtname = "Chiaroscuro"
```

Fig. 8. User query

```
select X, Y, Z,
       Xtname, Xdescription,
       Yaname, Yyear, Ypicture,
       Zcname, Zbiography
from {X:Technique}tname{Xtname},
     {X}exemplified_by{Y}.created_by{Z},
     {X}description{Xdescription},
     {Y}aname{Yaname},
     {Y}year{Yyear},
     {Y}picture{Ypicture},
     {Z}cname{Zcname}
     {Z}biography{Zbiography}
where Xtname = "Chiaroscuro"
```

Fig. 9. Query extension

3.5.2 Mediator

To be able to deal with disparate sources we adopt the idea of having a mediator, a component that enables the system to access several data sources reconciling their data in one coherent view; we apply this idea in the context of RDF and RQL. The mediator is responsible for finding the answer to the query by consulting the available sources based on the integration model instance. As shown in Figure 2 the mediator takes the extended query as its input. Then it proceeds as follows: for every variable occurring in the `select` clause of this query it locates an articulation(s) which contains this variable. From this articulation the mediator determines the name of the concept occurring in the source and also the way how to obtain that concept, i.e. the necessary transformer(s) for the concept values, the address of the source, and the path expression to the concept of interest within the source schema. This path expression can be seen as a query executed on a particular source. Hence, consulting articulations in the IM instance in fact means query unfolding as it is known in the GAV approach. The acknowledged disadvantage of the GAV approach is that in principle it requires changing the definition of the global schema (in our case the CM) each time a new source is added. This is however clearly not the case in our framework, since the only thing which changes when a new source is added or removed is the IM instance (new articulations are added or removed). From this point of view, we keep the CM independent from the sources, similarly to the LAV approach. Details concerning these approaches are beyond the scope of the paper and we refer the interested reader to a comparison presented in [27].

If there are more articulations found for a given variable, that means there are several competing sources offering values for this variable, the decorations attached to each articulation are used to decide the order in which the sources will be consulted.

After the sources are consulted, i.e. appropriate RQL queries are routed to them, the mediator waits for the response. Subsequently, it collects the results and assembles them into an answer which consists of a collection of tuples (in RDF terminology a bag of lists).

3.5.3 Result Extension

The answer provided by the mediator is a valid response to the (extended) RQL query that was asked, however it is not yet a CM instance. The result extension module transforms the "flat" collection of tuples by adding the appropriate properties into a valid RDF graph which adheres to the CM. This (query-dependent) CM instance serves as a basis for the presentation generation phase. Note that this phase actually re-constructs a part of the original RDF graph; this could have been omitted had the RQL language been able to produce as output again a full-fledged RDF graph. Our hope is that future RDF query languages will cover both the extraction and the construction part.

4 Presentation Generation

In the presentation generation phase the retrieved data is transformed in a hypermedia presentation suitable for the user platform and for the user preferences. The adaptive presentation generation (i.e. a presentation that considers the user's browsing history) will be explained in Section 5.

The presentation generation is composed from three steps: the application model generation, the application model instance generation, and the presentation data generation.

4.1 Application Model

The application model (AM) describes the navigational aspects of the hypermedia presentation. AM is composed of slices and slice properties that together define the navigation ontology. A slice is a meaningful presentation unit of some media items. These media items may originate from different CM concepts. There are two types of slice properties: slice composition, a slice encloses another slice, and slice navigation, a slice is the anchor of a hyperlink pointing to another slice. The most primitive slices are containing only one media item. Higher level slices will contain (using slice composition) other slices. At the top of the composition hierarchy are top-level slices which correspond to pages to be present on the user's display.

As depicted in Figure 3, the AM is represented in RDFS and there are two additional descriptions used in its definition: the AM properties (represented in RDFS) and the user/platform profile (represented in RDF). In addition to the user/platform profile, the AM can use also a user model. In adaptive applications the user model is responsible for storing information

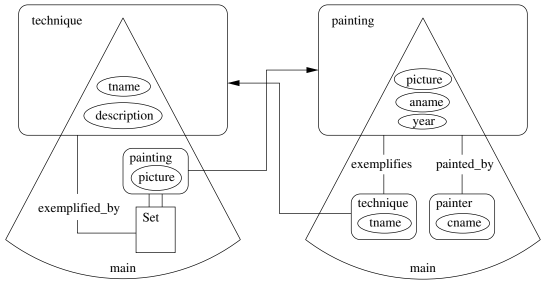

Fig. 10. Application model.

regarding the user browsing history. Modeling AM adaptivity using the user model will be explained in detail in Section 5.

The AM properties define the properties associated to the classes `Slice` and `Link`. The class `Slice` is the root of the navigation ontology. There are four types of slice properties: `owner` that associates a slice to a concept, `slice-ref` which denotes slice composition, `link` represents slice navigation, and `media` refers to the actual media. The `owner` property is the bridge between the AM and the CM that enables the transformation of CM query instances to AM instances. As `slice-ref` may connect slices belonging (based on their `owner` property) to different concepts, a `relationship-ref` property is attached to `slice-ref` to make explicit the concept relationship involved in the association. From the CM one can derive new concept relationships by composing the existing ones. Using these new concept relationships as a value for the `relationship-ref` property enables the embedding in the same slice of media items coming from concepts not directly linked in the CM. The class `Link` has two properties `source` and `destination` referring to the hyperlink anchor and hyperlink target respectively. Additionally, the AM defines the `SetOfSlices` and `SetOfLinks` classes to be used for one-to-many associations between concepts.

Figure 10 presents the AM for our running example. It defines a navigation ontology composed of two slices (subclasses of `Slice`) and two slice navigation properties between these two slices. The primitive slices are depicted as ovals and the slice composition properties are shown using nested composition notation. Since the relationship between `technique` and `painting` concepts is one-to-many we introduced a set of links when navigating from `technique` to `painting`.

In order to realize adaptation one can associate appearance conditions to slice references [19]. The appearance conditions enable two kinds of AM adaptation: *conditional inclusion of fragments* (slices in our context) and *link hiding* [3]. A link is hidden when its destination slice has an invalid condition. The slice appearance conditions use attribute-value pairs from the user/platform profile described below or from the user model described in Section 5. Conditions that use the user/platform profile elements (prefix `prf:`) specify *adaptability* and conditions that use the user model elements (prefix `um:`) specify *adaptivity*. Adaptability is done prior to the presentation browsing while adaptivity is done dynamically as the user model changes during the presentation browsing. The user/platform profile is static information (prior to presentation generation) while the user model represents dynamic information (generated on the fly as the user is browsing the presentation). Figure 11 gives an example of a condition for adaptability and two conditions for adaptivity.

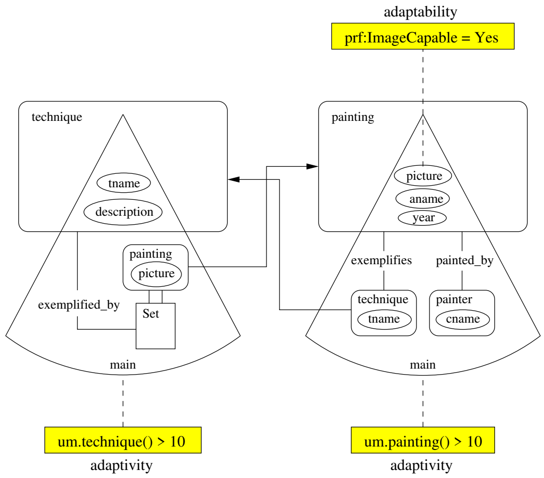

Fig. 11. Adaptation (adaptability/adaptivity) in the application model.

4.2 User/Platform Profile

The user/platform profile defines the device (display) capabilities and the user preferences. From Figure 3 one can see that the user/platform profile is an RDF description that instantiates two CC/PP vocabularies. As advocated in Section 2 one of the main advantages of RDF(S) is the ability to reuse existing RDFS vocabularies. UAProf is such a vocabulary developed by WAP Forum to model device capabilities (e.g. `ImageCapable` attribute). A new CC/PP vocabulary was created to model user preferences (e.g. `ExpertiseLevel` attribute).

4.3 Application Model Generation

In the application model generation step the AM is converted to an AM template. This step contains two substeps: the application model unfolding that generates the AM template and the application model adaptation that executes the adaptability specifications on the AM template.

4.3.1 Application Model Unfolding

In the application model unfolding the AM template is generated by an XSLT stylesheet. The AM template represents the structure of an AM instance (RDF) based on the AM schema (RDFS). Such a template will ease the specification of an XSLT stylesheet used to convert a CM instance to an AM instance. By unfolding the AM we mean repeating the process of adding properties inside the subject classes until slice references or media items are reached. In this way one obtains an AM template which will be filled later on with appropriate instances. In Figure 3 this substep is labeled 1.1.

4.3.2 Application Model Adaptation

The AM template needs to be adapted based on the specified slice appearance conditions. In this substep the AM adaptability is executed by an appropriate stylesheet. This stylesheet has two inputs: the AM template and the user/platform profile. The user/platform attributes are replaced in the conditions by their corresponding values. The slices that have the conditions not valid are discarded and the hyperlinks pointing to these slices are disabled. For the example depicted in Figure 11 the `picture` (primitive) slice will be suppressed for a user using a WAP phone (in the user/platform profile `prf:ImageCapable=No`). In Figure 3 this substep is labeled 1.2.

4.4 Application Model Instance Generation

In the application model instance generation the AM is instantiated with the retrieved data. This step is composed of two substeps: the application model instance transformation generation and the application model instance generation.

4.4.1 Application Model Instance Transformation Generation

The application model instance transformation generation builds the transformation stylesheet that will convert a CM instance to an AM instance. This step is using an XSLT stylesheet that will generate another XSLT stylesheet. One should note that an XSLT stylesheet is a valid XML file that can be produced by another XSLT stylesheet. This technique was also successfully used in the previous version of Hera which was XML-based [28]. The previously adapted AM template has all the information needed to specify such a transformation (remember the slice `owner` property that associates a slice to a concept). The implemented algorithm is straightforward: instantiate all slices for all the corresponding retrieved concept instances and each time a `slice-ref` is encountered refer to its identifier. We used the following name convention: a slice instance name (e.g. `Slice.painting.main_ID1`) is obtained from the slice name (e.g. `Slice.painting.main`) concatenated with the suffix (e.g. `ID1`) of the associated concept instance identifier (e.g. `Painting_ID1`). In Figure 3 this substep is labeled 2.1.

4.4.2 Application Model Instance Generation

In the application model instance generation the CM instance is converted to an AM instance. The XSLT stylesheet obtained in the previous substep is applied to the CM instance to yield an AM instance. As opposed to the previous transformations, this stylesheet will operate for inputs and outputs that are both query dependent. For each query Hera will dynamically instantiate the AM with the query result, i.e. a CM instance. In Figure 3 this substep is labeled 2.2.

4.5 Presentation Data Generation

The presentation data generation produces code specific for the user's browser. Figure 12 gives three snapshots of the hypermedia presentations generated for an HTML, WML, and SMIL browser. For each type of serialization a specific stylesheet is used. The stylesheets used for the HTML and SMIL use the ability of XSLT 2.0 [15] to generate multiple outputs.

In the code generation we used a media directed translation scheme: for each media type appropriate code is generated. For example, strings were represented in normal font and integers in italic font. For the WML browser images are not present and one may need to scroll down in order to view the full text. A back button similar to the back button from existing HTML/SMIL browsers was implemented for the WML serialization. In Figure 3 each of the alternatives for this step is having the same label 3.

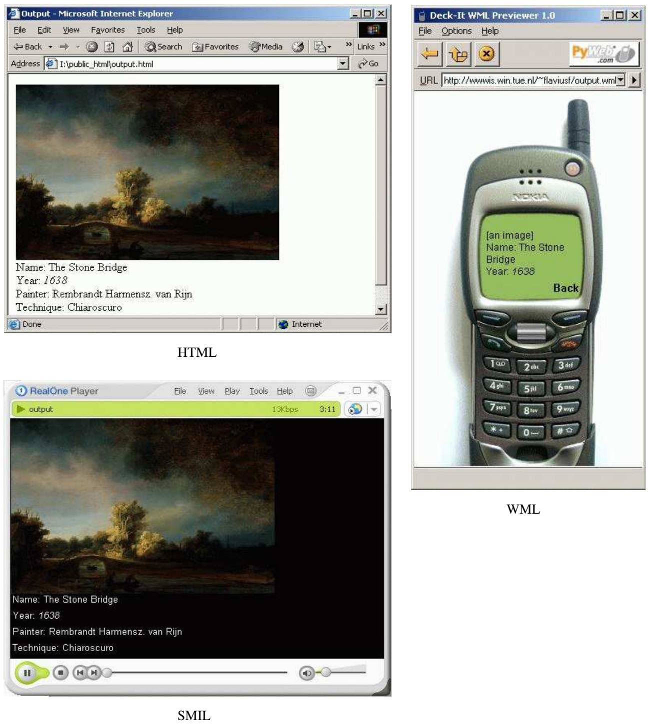

Fig. 12. Hypermedia presentation in different browsers.

5 AHA! Adaptivity Generation

In the previous section we have shown how to generate a hypermedia presentation based on an adaptable AM. The AM adaptability conditions use information from the device capabilities and user preferences stored in the user/platform profile. One can also generate adaptive hypermedia presentations by considering a user model that represents the user's state of mind based for example on the user's history of browsing this presentation.

Hera's adaptive features are designed in the spirit of AHAM (Adaptive Hypermedia Application Model) [22], a Dexter-based reference model for adaptive hypermedia. AHAM defines in the Storage Layer three models: the domain model, the user model, and the adaptation model. In Hera the above models have two parts: a model-based part and an instance-based part. The model-based part describes the model using schema elements (coming from CM/AM) that will be later on instantiated with data (coming from CM/AM instances).

In order to generate adaptive hypermedia presentations we can use the AHA! [29] system. Figure 13 gives an overview of how the AHA! software can be integrated into Hera. In the next three sections we will show how to build the domain model, the user model, and the adaptation model as input specifications for the AHA! system.

5.1 Domain Model

The domain model (DM) is composed of concepts/slices and their instances. Initially the domain model has all concepts/slices from the CM/AM. In response to a query the DM is automatically extended with data coming from the conceptual model instance and application

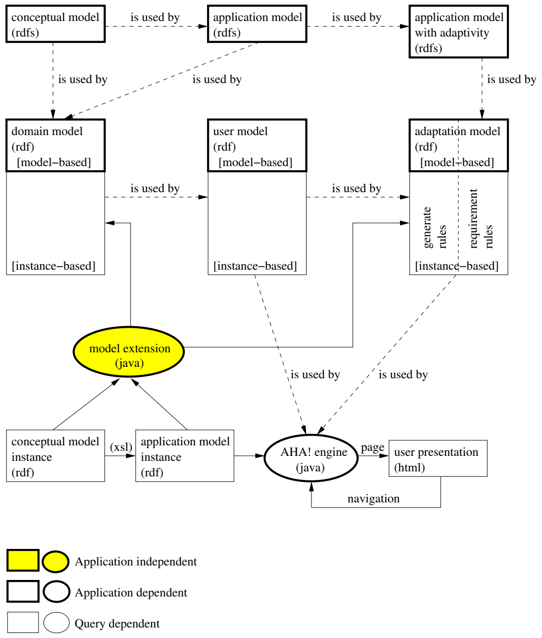

Fig. 13. Adaptivity in the presentation method.

model instance. Slice instances correspond to pages/fragments from AHAM. Top-level slice instances are similar to AHAM pages. The AHAM concept hierarchy is represented by the Hera conceptual model hierarchy having at the bottom slices (remember that slices are linked to concepts by the `owner` property). In this hierarchy, concepts and slices have `has instance` relationships to their instances.

5.2 User Model

The user model (UM) consists of a table of concept-value pairs. The UM concepts (called also AHA! concepts) originate from the previously defined DM and have an integer value between 0 and 100 associated. The generate rules from the adaptation model will modify the value of these AHA! concepts based on the user's browsing history. In the last version of AHA! (2.0) it is possible to associate multiple attributes (modeling different aspects of users) to an AHA! concept, but for the purpose of showing how to integrate AHA! in Hera one attribute is sufficient. Initially, before the first top-level slice is presented to the user all concepts from the UM have value 0.

5.3 Adaptation Model

There are two types of rules in the AHA! adaptation model: requirement rules and generate rules. Both rules are modeled at schema level. In response to a query, the adaptation model is automatically extended with data coming from the conceptual model instance and application model instance.

5.3.1 Requirement Rules

Requirement rules are defined by adaptivity conditions attached to slices. These conditions use information from the user model. A slice that has the condition valid is called a *desirable* slice. If the condition is not valid the slice is *not desirable.* Slices that are included in top-level slices and are not desirable are suppressed from the hypermedia presentation. Links pointing to top-level slices that are not desirable are disabled using the following mechanism. In AHA! a link pointing to a desirable slice is displayed in "good" color ("blue") if it was not visited before, "neutral" color ("purple") if it was visited before, and "bad" color ("black") if the link points to an undesirable slice. In Figure 11 two AM adaptivity conditions are presented. Note the use of `()` in `um:technique()` in order to differentiate the current technique instance from the general `um:technique` concept (from CM). Both conditions represent the desirability of these top-level slices based on the fact whether the user saw them before or not.

5.3.2 Generate Rules

Generate rules are UM update rules that are triggered when the user visits a top-level slice. When a desirable slice is visited its value is increased to 100. If the slice is not desirable its value is increased to 35. The initial increment is always a positive one. A change in the value of a slice will propagate to other elements from the UM based on the generate rules associated to that particular slice. These updates can be absolute or relative. Absolute updates propose a new (positive) value for the particular attribute. A relative update is a percentage of the increment of the AHA! concept in the rule header used for the updating of the AHA! concepts in the rule body. Relative updates can be positive or negative. Positive and negative relative updates are differentiated by the `+` and `-` prefixes respectively. All updates are rounded to the closest positive integer between 0 and 100. Updating AHA! concepts in a rule propagates the updating mechanism to the rules that have in their header the AHA! concepts mentioned in the body of the previous rule. In [29] sufficient conditions to ensure the termination of the propagation algorithm are proposed[h].

Figure 14 presents the generate rules for the two slices present in the AM. Suppose the hypermedia presentation starts with the slice instance of `slice.technique.main` corresponding to (owned by) the `technique` instance `Technique_ID1`. The value associated to this slice instance in the UM is updated to 100, representing that this slice has been visited. The first generate rule extension defines the consequences of this update. It states that the technique concept instance `Technique_ID1` and the associated five painting concept instances will have their values increased by $80\% \times 100 = 80$ and $20\% \times 100 = 20$ respectively. Because their initial value was 0 these increments will represent also the final value. Assume now that the user follows the painting instance `Painting_ID1` link pointing to a desirable slice (`um:painting(Painting_ID1)` $= 20 > 10$). The value of the slice instance of `slice.painting.main` corresponding to (owned by) the `painting` instance `Painting_ID1` is updated to 100. According to the second generate rule extension the painting concept instance `Painting_ID1`, the painter concept instance `Painter_ID1`, and the concept instance `Technique_ID1` have their values increased by $80\% \times 100 = 80$, $4\% \times 100 = 4$, and $1\% \times 100 = 1$ respectively. The value for `Painting_ID1` is updated to $20 + 80 = 100$, for `Painter_ID1` is updated to $0 + 4 = 4$, and for `Technique_ID1` is updated to $80 + 1 = 81$. The `technique` link

[h]Among these conditions we mention that the negative (relative) updates do not propagate.

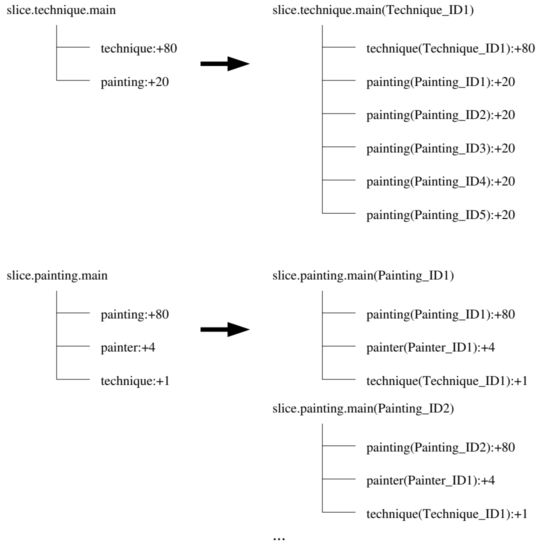

Fig. 14. Generate rule instantiation.

embedded in the slice instance points to a desirable slice (`um:technique(Technique_ID1)` $= 81 > 10$). As a consequence AHA! displays this link in "good" color.

Suppose now that the hypermedia presentation would have started from the slice instance of `slice.painting.main` corresponding to the `painting` concept instance `Painting_ID1`. The `technique` link points now to a undesirable slice (`um:technique(Technique_ID1)` $= 1 < 10$) because the user didn't receive yet enough information about the associated painting technique in order to be able to view the painting technique main slice. As a consequence AHA! displays this link in "bad" color. If the application model would permit the user to view more (than 10) paintings using the same painting technique, then the painting technique main slice would become a desirable slice (the value of `um:technique(Technique_ID1)` will increase with 1 at each view). In this case AHA! would display this link in "good" color.

6 Conclusions

Taking in consideration the Web evolution we extended the Hera methodology for the design of Semantic Web Information Systems. Such information systems make use of Semantic Web technology to support Web application interoperability. Hera is a model-driven methodology which uses different models (e.g. integration model, application model, adaptation model) for different aspects involved in the design of Web Information Systems, particularly the hypermedia aspects. This paper focuses on the design of the integration model and the adaptable/adaptive application model in order to support an automated process of generating adaptable/adaptive hypermedia presentations from different sources. As a Web ontology language is still in its infancy [30] we chose to represent Hera models in RDF(S) which is the foundation of the Semantic Web. In order to represent the different Hera models we provided appropriate RDF(S) extensions. The RDF/XML model serialization enabled the use of XSLT stylesheets as transformation specifications between the different model instances. This approach proved to be satisfactory if one does not need to use the RDF(S) inference rules in the transformation specification. As future work we plan to use (depending on their existence): a mature Web ontology language for representing the Hera models, a Web ontology-aware (or at least an RDF(S)-aware) transformation language to be used for the specification of the Hera transformations and an execution engine for this transformation language.

Acknowledgements

The authors would like to thank Rijksmuseum in Amsterdam for their kind permission to use copyrighted data in our example. We would like also to thank Lynda Hardman, Jacco van Ossenbruggen, Frank Nack, and Lloyd Rutledge from CWI, Amsterdam, and Paul De Bra from TUE, Eindhoven, for fruitful discussions. Part of the research described here has been founded by the NWO Dynamo project.

References

1. T. Isakowitz, M. Bieber, and F. Vitali (1998), *Web Information Systems*, Communications of the ACM, Vol. 41, No. 1, pp. 78–80.
2. S. Murugesan, Y. Deshpande, S. Hansen, and A. Ginige (2001), *Web engineering: a new discipline for development of Web-based systems*, in Web Engineering, Vol. 2016, Lecture Notes in Computer Science, Springer, pp. 3–13.
3. P. Brusilovsky (2001), *Adaptive hypermedia*, User Modeling and User-Adapted Interaction, Vol. 11, No. 1–2, pp. 87–110.
4. R. Vdovjak and G.J. Houben (2002), *Providing the semantic layer for WIS design*, in Advanced Information Systems Engineering, 14th International Conference, CAiSE 2002, Vol. 2348, Lecture Notes in Computer Science, Springer, pp. 584–599.
5. G. Mecca, P. Atzeni, A. Masci, P. Merialdo, and G. Sindoni (1998), *The Araneus Web-base management system*, in ACM SIGMOD International Conference on Management of Data, ACM, pp. 544–546.
6. M.F. Fernandez, D. Florescu, A.Y. Levy, and D. Suciu (2000), *Declarative specification of Web sites with Strudel*, VLDB Journal, Vol. 9, No. 1, pp. 38–55.
7. S. Ceri, P. Fraternali, and M. Matera (2002), *Conceptual modeling of data-intensive Web applications*, IEEE Internet Computing, Vol. 6, No. 4, pp. 20–30.
8. N. Koch, A. Kraus, and R. Hennicker (2001), *The authoring process of the UML-based Web engineering approach*, in First International Workshop on Web-Oriented Software Technology.
9. M. Cannataro, A. Cuzzocrea, C. Mastroianni, R. Ortale, and A. Pugliese (2002), *Modeling adaptive hypermedia with an object-oriented approach and XML*, in Second International Workshop on Web Dynamics.
10. Y. Jin, S. Xu, S. Decker, and G. Wiederhold (2002) *Managing Web sites with OntoWebber* in Advances in Database Technology, 8th International Conference on Extending Database Technology, EDBT 2002, Vol. 2287, Lecture Notes in Computer Science, Springer, p. 766.
11. R. Klapsing and G. Neumann (2000), *Applying the Resource Description Framework to Web engineering* in Electronic Commerce and Web Technologies, First International Conference, EC-Web 2000, Vol. 1875, Lecture Notes in Computer Science, Springer, pp. 229–238.
12. F. Frasincar, G.J. Houben, and R. Vdovjak (2002), *Specification framework for engineering adaptive Web applications*, in The Eleventh International World Wide Web Conference, Web Engineering Track, `http://www2002.org/CDROM/alternate/682/`.
13. O. Lassila and R.R. Swick (22 February 1999), *Resource Description Framework (RDF) model and syntax specification*, W3C Recommendation.
14. D. Brickley and R.V. Guha (23 January 2003), *RDF vocabulary description language 1.0: RDF Schema*, W3C Working Draft.

15. M. Kay (15 November 2002), *XSL Transformations (XSLT) version 2.0*, W3C Working Draft.
16. J. van Ossenbruggen, J. Geurts, F. Cornelissen, L. Hardman, and L. Rutledge (2001), *Towards second and third generation Web-based multimedia*, in The Tenth International World Wide Web Conference, ACM, pp. 479–488.
17. Wireless Application Protocol Forum Ltd. (2001), *Wireless application group: User Agent Profile*.
18. G. Klyne, F. Reynolds, C. Woodrow, and O. Hidetaka (8 November 2002), *Composite Capability/Preference Profiles (CC/PP): Structure and vocabularies*, W3C Working Draft.
19. F. Frasincar and G.J. Houben (2002), *Hypermedia presentation adaptation on the Semantic Web* in Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002, Vol. 2347, Lecture Notes in Computer Science, Springer, pp. 133–142.
20. G. Karvounarakis, V. Christophides, D. Plexousakis, and S. Alexaki (2001), *Querying RDF descriptions for community Web portals*, in 17iemes Journees Bases de Donnees Avancees, pp. 133–144.
21. J. Broekstra and A. Kampman (2001), *Sesame: A generic architecture for storing and querying RDF and RDF Schema*, Aidministrator Nederland.
22. P. De Bra, G.J. Houben, and H. Wu (1999), *AHAM: A Dexter-based reference model for adaptive hypermedia*, in The 10th ACM Conference on Hypertext and Hypermedia, ACM, pp. 147–156.
23. N. Koch and M. Wirsing (2002), *The Munich Reference Model for adaptive hypermedia applications*, in Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002, Vol. 2347, Lecture Notes in Computer Science, Springer, pp. 213–222.
24. R. Vdovjak and G.J. Houben (2001), *RDF-based architecture for semantic integration of heterogeneous information sources*, in International Workshop on Information Integration on the Web.
25. N.F. Noy and M.A. Musen (2001), *Anchor-prompt: Using non-local context for semantic matching*, in Workshop on Ontologies and Information Sharing at the Seventeenth International Joint Conference on Artificial Intelligence, 2001.
26. R. Vdovjak, P. Barna, and G.J. Houben (2003), *EROS: explorer for RDFS-based ontologies*, in ACM International Conference on Intelligent User Interfaces, IUI 2003, ACM, pp. 330.
27. J.D. Ullman (1997), *Information integration using logical views*, in Proceedings of the 6th Int. Conference on Database Theory, ICDT 1997, Vol. 1186, Lecture Notes in Computer Science, Springer, pp. 19–40.
28. F. Frasincar and G.J. Houben (2001), *XML-based automatic Web presentation generation*, in WebNet 2001 World Conference on the WWW and Internet, AACE, pp. 372–377.
29. P. De Bra, A. Aerts, G.J. Houben, and H. Wu (2000), *Making general-purpose adaptive hypermedia work* in WebNet 2000 World Conference on the WWW and Internet, AACE, pp. 117–123.
30. F. van Harmelen, J. Hendler, J. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L.A. Stein (2003), *OWL Web ontology language 1.0 reference*, W3C Working Draft.