Engineering Semantic Web Information Systems

Richard Vdovjak, Flavius Frasincar, Geert-Jan Houben, and Peter Barna

Department of Computer Science, Technische Universiteit Eindhoven,
PO Box 513, Eindhoven, the Netherlands
{richardv,flaviusf,houben,pbarna}@win.tue.nl

Abstract

Web Information Systems (WIS) use the Web paradigm and technologies to retrieve information from sources connected to the Web, and present the information in a web or hypermedia presentation to the user. Hera is a design methodology that supports the design of WIS. It is a model-driven method that distinguishes integration, data gathering, and presentation generation. In this paper we address the Hera methodology and specifically explain the integration model that covers the different aspects of integration, and the adaptation model, that specifies how the generated presentations are adaptable (e.g. device capabilities, user preferences). The Hera software framework provides a set of transformations that allow a WIS to go from integration to presentation generation. These transformations are based on RDF(S), and we show how RDF(S) has proven its value in combining all relevant aspects of WIS design. In this way, RDF(S) being the foundation of the Semantic Web, Hera allows the engineering of Semantic Web Information Systems (SWIS).

Keywords

WIS, SWIS, RDF(S), Semantic Web, XSLT, Hera

Introduction

On the basis of its number of users and the attention it attracts, it is fair to say that the World Wide Web is the most popular source of information. With its overwhelming success and its considerable influence on the way in which we exchange information, some compare it to Gutenberg's invention of the printing press. Computer applications make information available for a very diverse audience on different platforms worldwide and 24 hours a day. In this context of the Web, the nature of information systems has changed.

Early applications on the Web were mainly applications that presented data in terms of carefully authored hyperdocuments. The author hand-crafted a static collection of pages and links between these pages in order to convey information to the users. Since more and more data sources get connected to the Web, more information has become available and we see that the typical Web application is now data-intensive. The application uses for example data generated from databases in order to deliver the right information to the user. This trend has changed the notion of information system in the sense that a modern professional information system has become a web application: a Web Information System (WIS), as introduced by Isakowitz et al. (1998), uses the Web paradigm (and technologies) to retrieve information from the available sources and deliver it to the users.

Typical for the spirit of the Web is 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 applications not applicable anymore. Developing a hyperdocument typically meant ad-hoc programming and a mix of content and presentation design. The data-intensive nature of a WIS requires a more rigorous development process. One reason is that the one-size-fits-all approach 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).

This paper follows up on the model-driven approach of Hera described in Frasincar at al. (2002) 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) (Lassila and Swick 1999; Brickley and Guha 2003) 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, allowing this presentation generation software to be modularly combined with other software and/or data. Moreover, the paper specifies how ontologies expressed in RDF(S) help in the entire process of integrating and retrieving data and generating presentations. With the central role for RDF(S) we are building first Semantic Web Information Systems (SWIS).

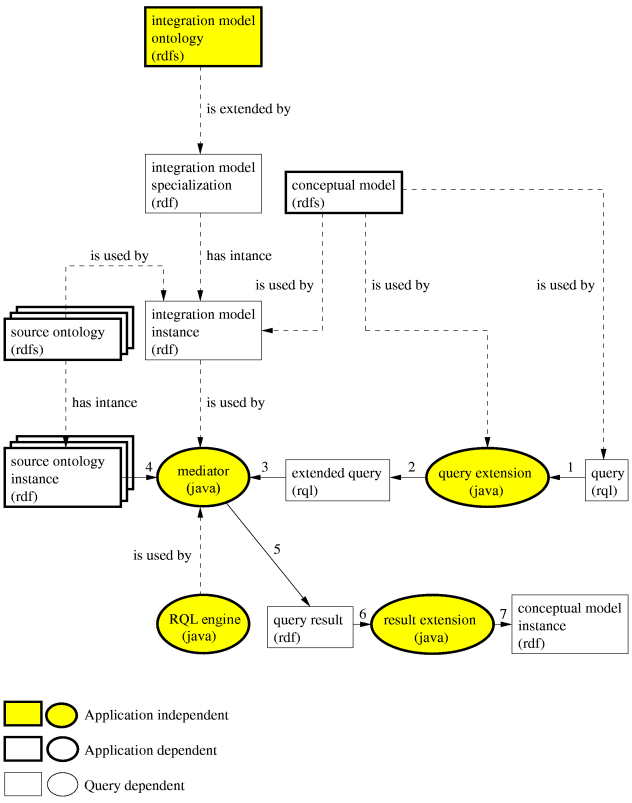


Figure 1. Integration and data gathering

Hera Methodology

The main target of the Hera project is to provide support for WIS design and implementation. A typical WIS generates a hypermedia presentation from data that is gathered as a response to a user query from several, possibly heterogeneous, data sources. This entire process of retrieving data and presenting it in hypermedia format needs to be specified during the design phase and the WIS has to be programmed in such a way that it can automatically execute that specification.

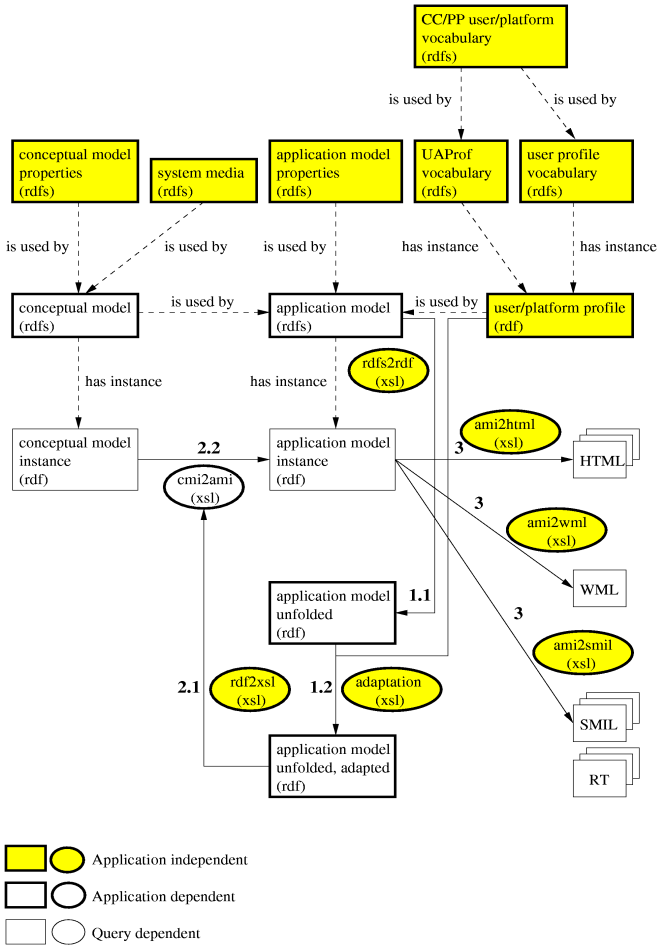


Figure 2. Presentation generation

Hera's approach adopts the principle of separation of concerns and builds the application model on top of the conceptual model. This facilitates model-driven transformations to populate the model of the application with the retrieved data. The Hera methodology distinguishes two principal phases each consisting of several different data transformations that are necessary to generate the hypermedia output in response to a user query.

- integration and data gathering (depicted in Figure 1)
- presentation generation (depicted in Figure 2)[1]

[1] 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.

RDF(S) is the main format used in both of these phases. 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 effectively enables web application interoperability.

Related Work

Most of the web engineering approaches do not consider adaptation, as opposed to what is the case in our Hera methodology. We note that models like AHAM (De Bra et al. 1999) or MRM (Koch and Wirsing 2002) consider adaptation, but only in the context of adaptive hypermedia documents: the generation of hypermedia presentations in WIS poses different requirements. As notable exceptions among the web engineering approaches we mention XAHM (Cannataro et al. 2002), an XML-based methodology, UWE (Koch et al. 2001), 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) (Klapsing and Neumann 2000) 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.

Integration and Data Gathering

The main task of this phase is to connect the conceptual model with several autonomous sources by creating channels through which on request the concepts from the conceptual model will be populated with data. 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.

Running Example

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 (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 conceptual model.

The conceptual model (CM) provides a uniform semantic view over multiple data sources. The CM serves as an interface between data gathering 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.

Figure 3 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.

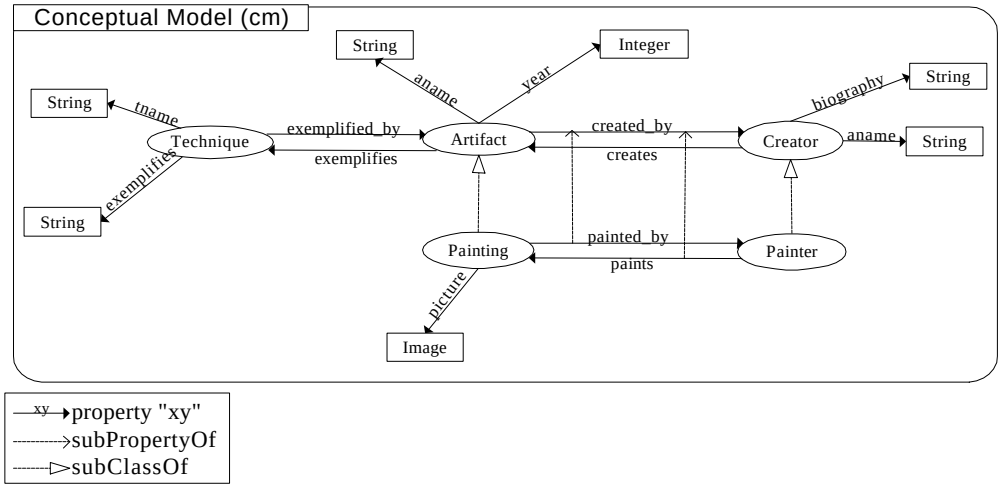


Figure 3. Conceptual model

Integration

The Hera framework provides the designer with an integration template that takes a form of a predefined integration model / ontology. During the integration phase the designer tailors this predefined model by extending it to meet the application specifics. This model is subsequently instantiated. This instantiation effectively establishes mappings between the concepts from the CM and those from the data sources. The process of specializing and instantiating the integration model is performed only once, prior to the user asking the query. Figure 1 (top) depicts the sequence of the design steps involved in the integration phase.

In our previous work (Vdovjak and Houben 2001, 2002) 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. Disregarding the issue of syntactical differences, 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 (Karvounarakis et al. 2001) query services.

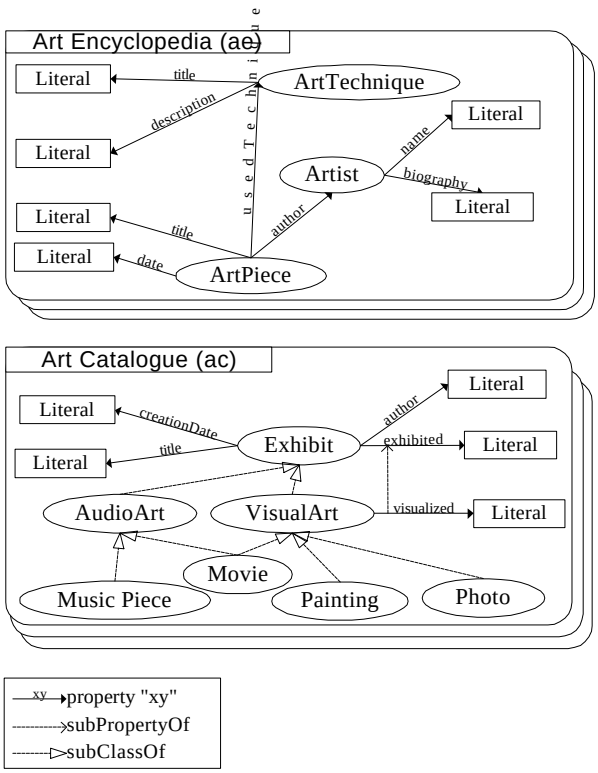


Figure 4. Integrated sources

It is often the case on the Web that the information is duplicated and offered 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.

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 4. 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. Note the abbreviated names in parentheses, which denote different namespaces that later will uniquely identify the sources in the integration model.

Integration Model

The integration model (IM) addresses the problem of relating concepts form the source ontologies to those from the CM. This problem can also be seen as the problem of merging or aligning ontologies. 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.

The integration model ontology (IMO) depicted in Figure 5 is a meta-ontology 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*.

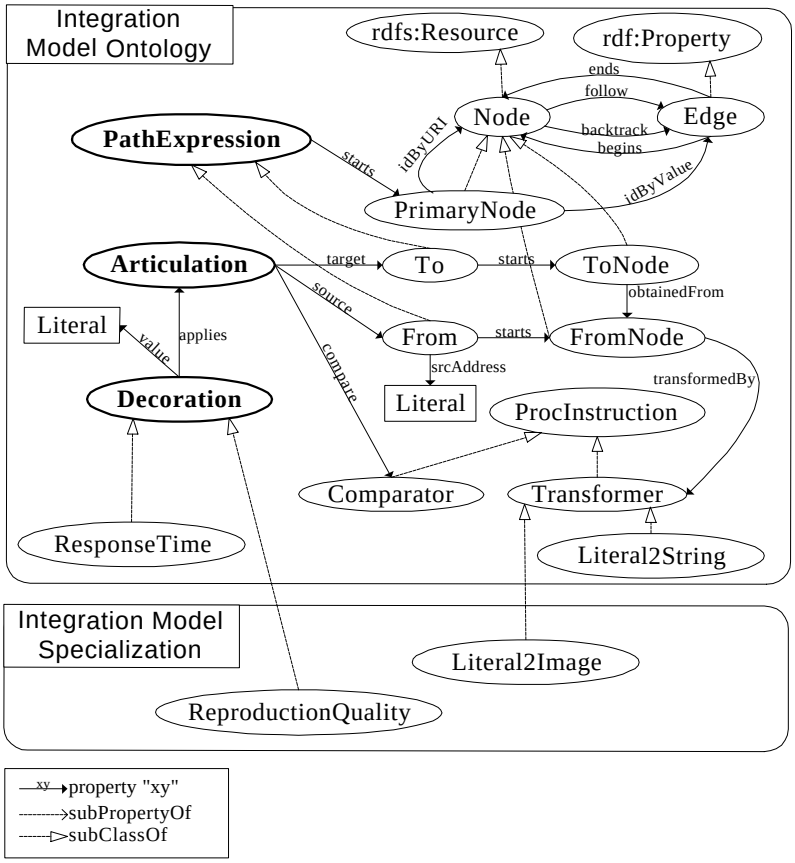


Figure 5. Integration model ontology and a specialization

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 supposed to be extended 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 5 (bottom).

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* 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 of which serves as an ID.

Each 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* that extends the *Node* with two properties: *obtainedFrom* and *producedBy*. 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).

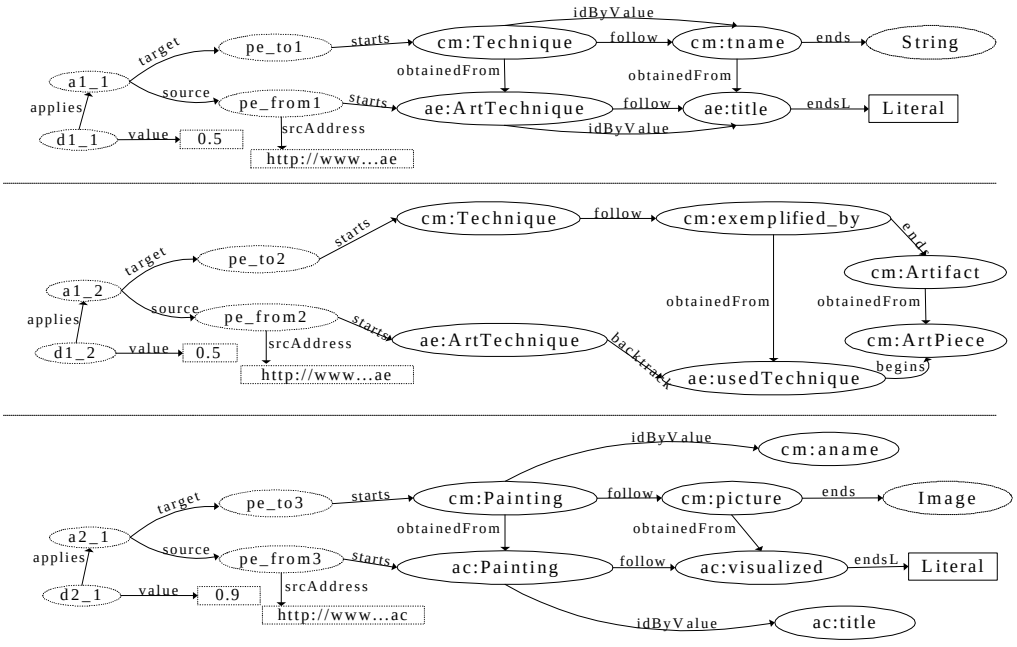


Figure 6. Articulations in the integration model instance

Integration Model Instance

The integration model instance is produced by the designer by instantiating the IMO. 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 their instances. Figure 6 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 called *ae:usedTechnique*. This relationship is reached in the *from* path by means of the aforementioned *backtrack* meta-property.

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.

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

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

Figure 7. User query and its extension

Data Gathering

While the integration phase (instantiating the IM) is performed only once, prior to the user asking the query, the data gathering 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 1 shows the dataflow of this phase with three processing blocks involved: the query extension, the mediator, and the result extension.

In the sub-phase of query extension the RQL user query, an example of which is depicted in Figure 7 (top), 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. Figure 7 (bottom) depicts the result of the query extension for the mentioned user query.

In the sub-phase of data mediation the mediator finds an answer to the extended query by consulting the available sources based on the integration model instance. The mediator takes the extended query as its input and 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 Global-As-View (GAV) approach.

However, as opposed to GAV that requires changes in the definition of the global schema each time a new source is added or removed, our framework in this scenario changes only the IM instance (new articulations are added or removed). From this point of view, we keep the CM independent from the sources, similarly to Local-As-View (LAV). Details concerning GAV and LAV approaches are beyond the scope of the paper and we refer the interested reader to a comparison presented in Ullman (1997).

If there are more articulations found for a given variable, that means there are several competing sources offering values for this variable. In this case 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 that consists of a collection of tuples (in RDF terminology a bag of lists).

The last subphase is the result extension. The answer provided by the mediator is a valid response to the 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 that adheres to the CM. This (query-dependent) CM instance serves as a basis for the presentation generation phase.

Presentation Generation

In the presentation generation the retrieved data is transformed in a hypermedia presentation suitable for the user platform and for the user preferences. The presentation generation is composed from three steps: the application model generation, the application model instance generation, and the presentation data generation.

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 a media item. Higher-level slices contain (using slice composition) other slices. At the top of the composition hierarchy are top-level slices that correspond to pages to be present on the user's display.

The AM distinguishes several 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.

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 8 presents the AM for our running example. It defines a navigation ontology composed of two slices and two slice navigation properties. Since the relationship between *technique* and *painting* concepts is one-to-many we introduced a set of links when navigating from *technique* to *painting*.

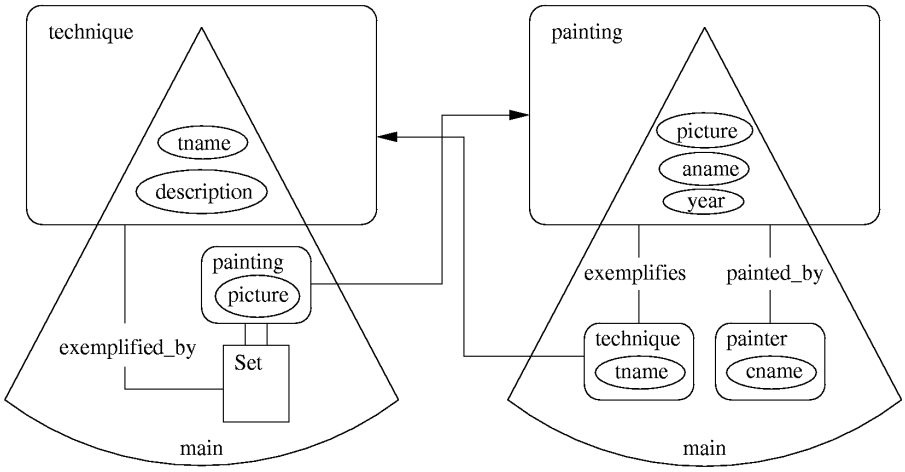


Figure 8. Application model

In order to realize adaptation one can associate appearance conditions to slice references (Frasincar and Houben 2002). The appearance conditions enable two kinds of AM adaptation: *conditional inclusion of fragments* (slices in our context) and *link hiding* (Brusilovsky 2001). 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 details of which are outside of the scope of this paper. 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[2]. 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 9 gives an example of a condition for adaptability and two conditions for adaptivity.

[2] Here we mainly focus on adaptability leaving the details concerning adaptivity out of scope of this paper.

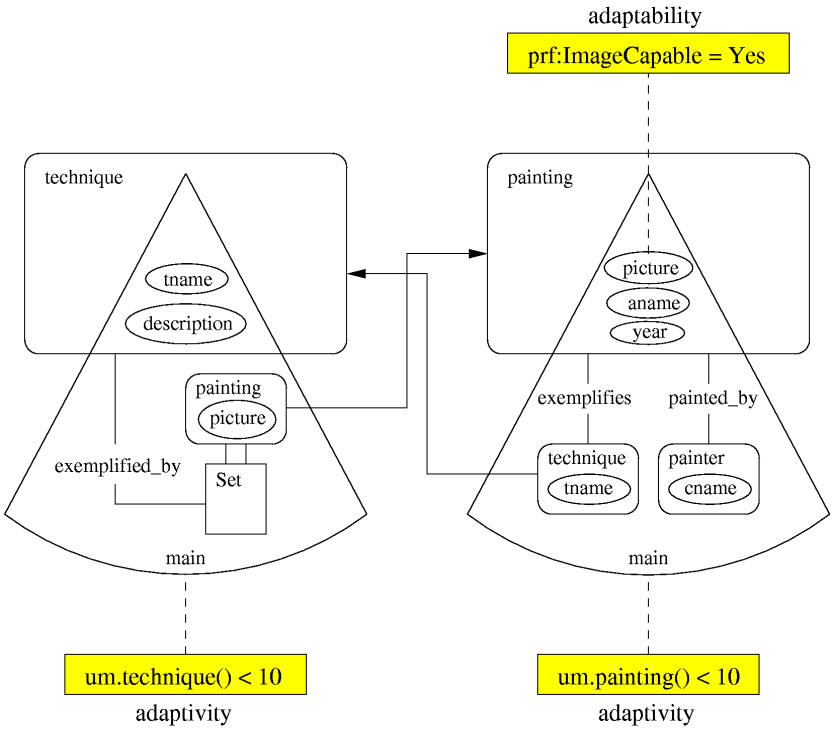


Figure 9. Adaptation (adaptability/adaptivity) in the application model

The user/platform profile defines the device (display) capabilities and the user preferences. From Figure 2 one can see that the user/platform profile is an RDF description that instantiates two CC/PP vocabularies. 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).

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.

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 that will be filled later on with appropriate instances.

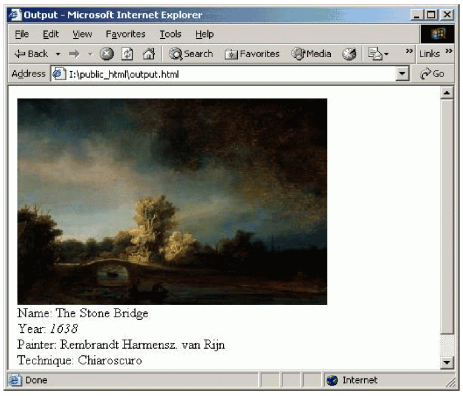
The AM template needs to be adapted based on the specified slice appearance conditions. In this substep the AM adaptability is executed by an appropriate stylesheet. This stylesheet has two inputs: the AM template and the user/platform profile. The user/platform attributes are replaced in the conditions 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 9 the *picture* (primitive) slice will be suppressed for a user using a WAP phone (in the user/platform profile *prf:ImageCapable=No*).

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.

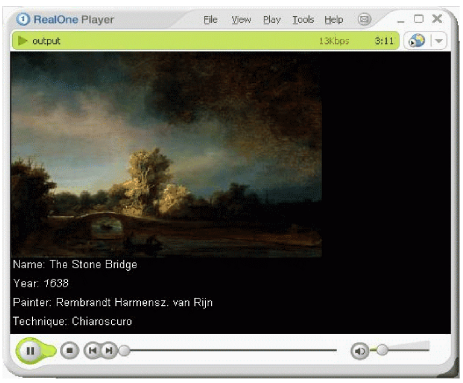
The application model instance transformation generation builds the transformation stylesheet that will convert a CM instance to an AM instance. This step is using an XSLT stylesheet that will generate another XSLT stylesheet. One should note that an XSLT stylesheet is a valid XML file that can be produced by another XSLT stylesheet. This technique was also successfully used in the previous version of Hera which was XML-based (Frasincar and Houben 2001). 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 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.



HTML

WML



SMIL

Figure 10. Hypermedia presentation in different browsers

Presentation Data Generation

The presentation data generation produces code specific for the user's browser. Figure 10 gives three snapshots of the hypermedia presentations generated for a 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 (Kay 2002) 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.

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 that uses different models (e.g. integration model, application model, adaptation model) for different aspects involved in the design of web information systems. 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 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.

References

Brickley, D & Guha, RV (2003), 'RDF Vocabulary Description Language 1.0: RDF Schema', W3C Working Draft, http://www.w3.org/TR/rdf-schema/.

Brusilovsky, P (2001), 'Adaptive Hypermedia', *User Modeling and User-Adapted Interaction*, vol. 11, no. 1-2, pp. 87-110.

Cannataro, M, Cuzzocrea, A, Mastroianni, C, Ortale, R & Pugliese, A (2002), 'Modeling Adaptive Hypermedia with an Object-Oriented Approach and XML' in *Proceedings of the Second International Workshop on Web Dynamics*.

De Bra, P, Houben, GJ & Hongjing, W (1999), 'AHAM: A Dexter-based Reference Model for Adaptive Hypermedia' in *Proceedings of the 10th ACM Conference on Hypertext and Hypermedia*, pp. 147-156.

Frasincar, F & Houben GJ (2001), 'XML-Based Automatic Web Presentation Generation' in *WebNet 2001 World Conference on the WWW and Internet*, pp. 372-377.

Frasincar, F, Houben, GJ & Vdovjak, R (2002), 'Specification Framework for Engineering Adaptive Web Applications', *The Eleventh International World Wide Web Conference, Web Engineering Track*, http://www2002.org/CDROM/alternate/682/

Frasincar, F & Houben GJ (2002), 'Hypermedia Presentation Adaptation on the Semantic Web' *in proceedings of the Adaptive Hypermedia and Adaptive Web-Based Systems Second International Conference, (AH'02),* vol. 2347, pp. 133-142.

Isakowitz, T, Bieber, M & Vitali, F (1998), 'Web Information Systems', *Communications of the ACM*, vol. 41, no. 1, pp. 78-80.

Karvounarakis, G, Christophides, V, Plexousakis, D, Alexaki, S (2001), 'Querying RDF Descriptions for Community Web Portals' in *17iemes Journees Bases de Donnees Avancees,*pp. 133-144.

Kay, M (2002), 'XSL Transformations (XSLT) Version 2.0', W3C Working Draft, http://www.w3.org/TR/xslt20/.

Klapsing, R & Neumann, G (2000), 'Applying the Resource Description Framework to Web Engineering' in *Proceedings of the First International Conference on Electronic Commerce and Web Technologies,* vol. 1875, pp. 229-238.

Koch, N, Kraus, A & Hennicker, R (2001), 'The Authoring Process of the UML-based Web Engineering Approach' in *Proceedings of the First International Workshop on Web-Oriented Software Technology.*

Koch, N & Wirsing, M (2002), 'The Munich Reference Model for Adaptive Hypermedia Applications' in Proceedings of the Adaptive Hypermedia and Adaptive Web-Based Systems Second International Conference, vol. 2347, pp. 213-222.

Lassila, O & Swick, R.(1999), 'Resource Description Framework (RDF) Model and Syntax Specification', W3C Recommendation, http://www.w3.org/TR/REC-rdf-syntax/.

Ullman, JD (1997), 'Information integration using logical views' *in Proceedings of the 6th International Conference on Database Theory, (ICDT'97),* vol. 1186, pp. 19-40.

Vdovjak, R & Houben GJ (2002), 'Providing the Semantic Layer for WIS Design' in Proceedings of the 14th International Conference on Advanced Information Systems Engineering, (CAiSE'02), vol. 2348, pp. 584-599.

Vdovjak, R & Houben, GJ (2001), 'RDF-Based Architecture for Semantic Integration of Heterogeneous Information Sources' Design' *in Proceedings of the International Workshop on Information Integration on the Web.*