

Adaptive Hypermedia Presentation Modeling for Domain Ontologies

José Antonio Macías and Pablo Castells

E.T.S. de Informática, Universidad Autónoma de Madrid
Campus de Cantoblanco, 28049 Madrid, Spain
+34-91-3482241, +34-91-3482284
{j.macias, pablo.castells}@uam.es
<http://www.ii.uam.es/{~macias, ~castells}>

Abstract. Adaptive hypermedia has become a hot topic as web-based on-line instruction keeps thriving. We propose a framework that supports the definition of dynamic hypermedia presentations without assuming a pre-established approach for pedagogical knowledge representation. In our system authors a) define ontologies that best fit a specific domain or the author's understanding of the domain, and b) design abstract presentation models associated to knowledge categories. Web pages are generated dynamically from the domain and presentation model, with a strong correspondence between domain structure and the derived hyperspace structure. The system provides explicit support for adaptive presentation constructs, and admits external adaptive navigation mechanisms and user model update strategies.

1. Introduction

The automatic adaptation of hypermedia-based educational systems to student characteristics and performance has become a first-order concern as web-based technology is taking up an increasingly important role in the educational field. To date, most research efforts in adaptive hypermedia have focussed on course sequencing and content selection [Brusilovsky 1998], while less attention has been paid to presentation. Support for adaptive presentation in existing tools include aspects like link style and annotation, choice of style sheets for documents, contents selection, and spatial ordering of text and media fragments. Usually these aspects are partly defined in detail by the designer for each learning unit (as in [De Bra 1999]), and partly set up automatically by the system according to fixed presentation templates, common to all units, that the designer cannot configure (see [Vassileva 1997, Carro 1999] for instance). Our work aims at filling this gap by providing courseware designers with a higher degree of control over the generation of all aspects of presentation, including adaptive capabilities, at a moderate development cost.

The goal of our research is to support the design of adaptive hypermedia presentations in such a way that a) course presentation can be specified independently from course contents, b) non-trivial presentation constructs involving relations between domain objects can be generated from very succinct high-level descriptions, and c) the presentation system is general enough to be compatible with a reasonably wide variety of domain models. While our research is based on observations taken mainly from the educational field, no essential assumptions have been made as of yet that would prevent it from being applied to other domains.

Our approach consists of:

1. Supporting the definition of made-to-measure domain ontologies for the description and conceptual structuring of subject matter (as in [Murray 1998]).
2. Describing the structure and contents of specific courses in terms of the vocabulary defined by the ontology.
3. Defining adaptive presentation templates and presentation rules associated to specific terms (classes, relationships and/or class instances) of the ontology.

2. Domain Ontology

An ontology is a shared conceptual representation of domain knowledge that provides a common understanding of a domain. Ontologies were originally developed in Artificial Intelligence to facilitate knowledge sharing and reuse [Gruber 1993]. Later, ontologies have been used for intelligent knowledge retrieval in the WWW as an instrument to model semantic information (metadata) used to annotate web documents (see e.g. [Fensel 1999]). In our approach,

ontologies are used to provide maximum flexibility in the representation of pedagogical knowledge. Moreover, they are an essential element in achieving the separation of presentation and contents [Macías 2001].

In our system, ontologies consist of a set of classes that best suit the nature of a specific domain or that reflect the particular vision of a specific author on the domain. Ontologies can be defined with a high degree of freedom. Classes can be very generic, like `Concept` or `LearningUnit`, or reflect a finer-grained approach, like `Algorithm`, `Definition` or `Theorem`, as the designer sees fit. Ontologies include terms for subject matter information (e.g. a theorem has a statement and a proof), pedagogical information (e.g. lessons have levels of difficulty), and run-time state information (e.g. whether a concept is known by the user). All this knowledge is captured by defining attributes for classes, and relations between classes. Two predefined root classes are provided: `Topic` and `Fragment`, for ontology designers to subclass. `Topic`'s and `Fragment`'s are different in that the former are presented to the end-user in a separate page, while fragments can be inserted in a page together with other fragments and links to topics. A predefined subclass of `Fragment`, `AtomicFragment`, is also provided that consists of HTML code, either in the form of a literal string, or as a URL from which HTML contents are to be retrieved. In addition to a domain ontology, simpler data structures are defined by the designer to describe user profiles, platform characteristics, and other aspects considered relevant for adaptive presentation.

Courses are constructed by creating instances of ontology classes and setting relations between instances, building semantic networks of topic and fragment subclasses, where multimedia contents are included as atomic fragments. Though we are completing the development of interactive authoring tools for the domain model, ontology classes and instances are currently described textually in XML documents. For instance, assuming classes like `Painter`, `Artwork`, and `ArtisticStyle` have been defined for a course on Art History, the following example shows a simplified version of an object of class `Painter` that represents knowledge about van Gogh.

```
<Painter id="vangogh" name="Vincent van Gogh"
  birth="1853" death="1890" nationality="Dutch">
  <school> <ArtisticStyle ref="postimpressionism"/> </school>
  <picture> <AtomicFragment url="vangogh-picture.jpg"/> </picture>
  <shortIntro>
    <AtomicFragment> Generally considered the greatest Dutch painter after
    Rembrandt, he powerfully influenced the current of Expressionism in modern
    art. His work is characterized by the striking colour, coarse brushwork,
    and contoured forms. Among his masterpieces are numerous self-portraits
    and the well-known The Starry Night (1889). </AtomicFragment>
  </shortIntro>
  <biography> <AtomicFragment url="vangogh-bio.html"/> </biography>
  <works>
    <Artwork ref="starrynight"/>
    <Artwork ref="sunflowers1"/>
    <Artwork ref="irises"/>
  </works>
</Painter>
```

XML attributes like `name` and `birth` stand for domain object properties, whereas `school`, `picture`, `shortIntro`, `biography` and `works` are relationships with other domain objects (the `ref` attribute indicates referenced object ID's). Literal fragments can be inserted inline as XML elements (like `shortIntro`), or stored in external files (web addresses) that are referenced in the XML code (like `picture` and `biography`).

In addition to capturing domain semantics, ontology instances (domain objects) like the one above are used to maintain an overlay model of student knowledge with respect to each domain item. We assume that this model is dynamically updated to reflect the progress of the user and his/her changing state of mind as s/he pursues the course. The topic network itself may also be dynamically set up and modified at runtime. Our presentation system takes care of how the presentation should react to state changes, but how course structure, user model, and application state are updated is external to the presentation system.

3. Presentation Model

Presentations are defined by creating a template for each class of the ontology. Templates are defined using a textual language based on JavaServer Pages™ [Sun 2001], that allows the presentation designer to insert Java expressions (between `<%=` and `%>`) and control statements (between `<%` and `%>`) into HTML code. A template defines what parts

(attributes and relations) of a topic must be included in its presentation, their visual appearance and layout. For example, a simple template for the class Painter can be defined as follows:

```
<center>
<h2> <%= name %> </h2>
(<%= nationality %>, <%= birth %> - <%= death %>) <br>
</center><br>
<center><table>
<tr><td valign="top" rowspan="5"> <%= picture %> </td>
<td valign="top"> <%= shortIntro %> </td></tr>
<tr><td> <%= biography %> </td></tr>
<tr><td> <%= works %> </td></tr>
<tr><td> <%= school %> </td></tr>
</table></center>
```

Dynamic presentation constructs are generated from simple descriptions at a high level of abstraction. In the example above, to include information about important works of art in a painter's page, the designer only needs to refer to the `works` relation for the displayed object (shown in bold). The system automatically takes care of deciding whether to insert the corresponding works into the generated page, to generate a link for each one, or a single link for all of them, which style and/or visual cues are applied in the latter cases, and how all the pieces are laid out. In doing so, the system analyzes whether the relation is simple or multivalued, the class of the involved topics or fragments, their state, and other conditions, if any, stated by the designer. Figure 1 shows the web page generated using this presentation template for the "van Gogh" domain unit described in the previous section.

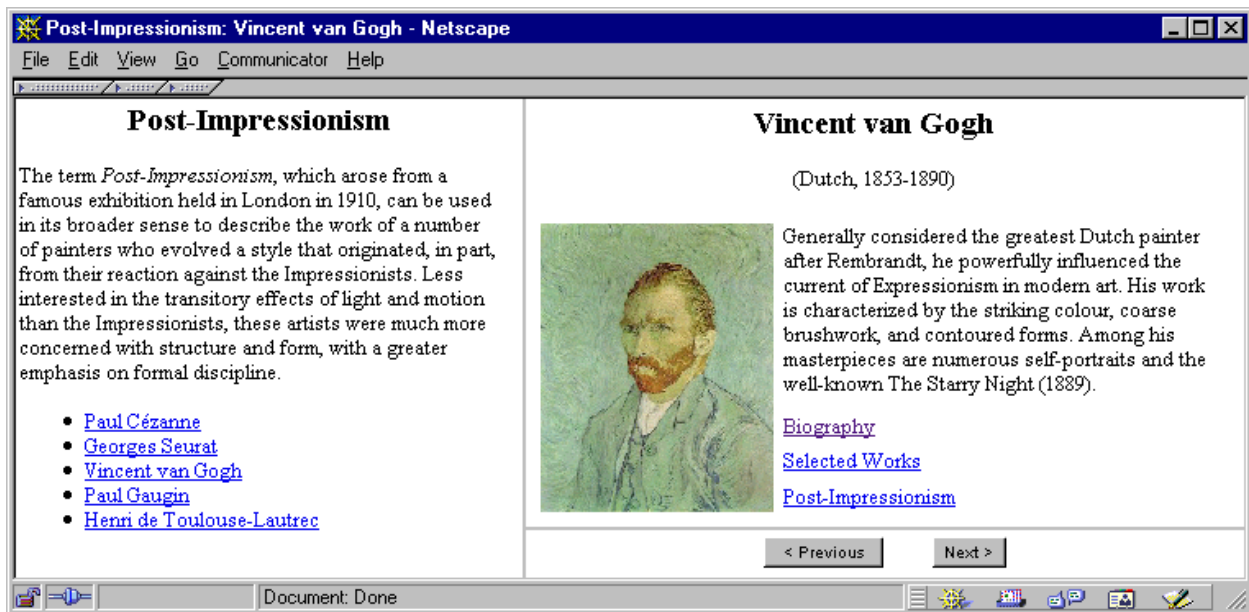


Figure 1. Web page generated for a knowledge item of class Painter

How presentation choices are inferred from conditions like the ones mentioned in the previous paragraph is established by rules that the designer can customize. For instance, the following rule is responsible for the generation of a link to van Gogh's biography rather than inserting the full text because the length of the text exceeds a certain threshold.

```
<Rule class="AtomicFragment">
  <test condition="length > 500"/>
  <presentation> <%= this.asLink(true) %> </presentation>
</Rule>
```

This rule could be made more precise by adding, for instance, conditions on the interaction preferences of the user. By way of another rule, a single link to a painter's works is created instead of a list of links when the number of items is greater or equal than three (we have intentionally set a small value here for the purpose of demonstration):

```

<Rule class="List[KnowledgeUnit]">
  <test-every var="item" list="this" condition="item.asLink"/>
  <test condition="count > 3"/>
  <presentation> <a href="<%= generateURI(this) %>"> <%= title %> </a> </presentation>
</Rule>

```

Other rules can be used to implement adaptive topic link annotation techniques, by setting link styles according to student knowledge state with respect to topics, as is usual in adaptive hypermedia support systems [Brusilovsky 1998]. The scope of presentation rules can be a class, a relation, a class instance, or a combination of them, or the whole course.

Adaptivity is achieved by putting conditions on templates, on parts inside templates, in presentation rules, and over relations between domain objects. These conditions can test properties of the user model (overlay model and user profile), properties of the data, characteristics of the platform, and any other aspect that should influence presentation, like course requirements, student's goals, usage modes (e.g. learning vs. consultation), etc.

At runtime, when the student moves to a course topic by traversing a link, the topic instance and its matching template are found in the server, from which an HTML page is generated automatically and sent back to the student's web browser. Before applying the template, all applicable rules are fired on the topic instance (see Figure 2). In general, templates and rules produce unfinished documents in the sense that the output document still contains other referenced domain objects that must be presented by recursively applying templates and rules. If several templates have been defined along the ontology class hierarchy, child-class templates are nested in parent-class templates. In particular, consistent course presentation framings can be easily constructed by defining templates for predefined classes at the top of the hierarchy. For example, HTML elements surrounding the painter presentation in Figure 1 (frame structure with contextual index on the left and *Previous / Next* buttons at the bottom) come from the presentation template for the parent class Topic. The generation process is complete when a document results that contains no remaining unprocessed objects.

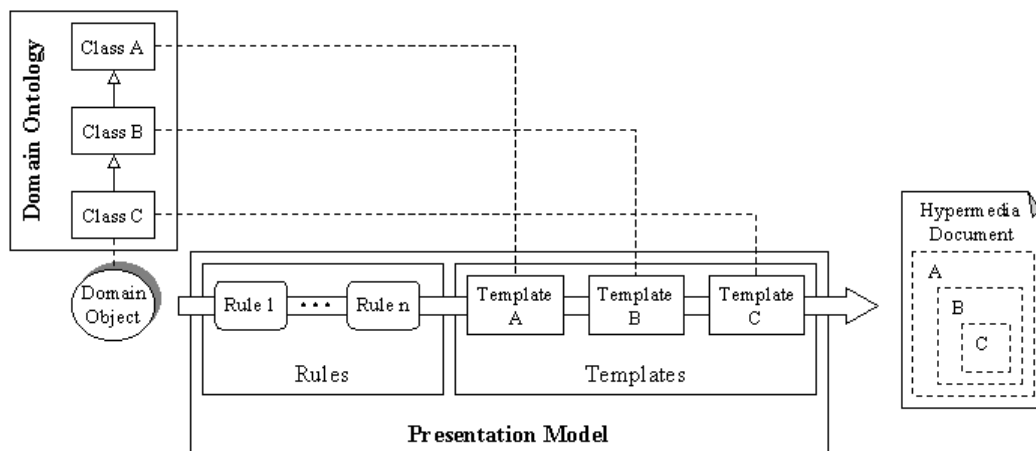


Figure 2. Hypermedia document generation from domain objects by applying presentation templates and rules

In our system, the unit of interaction with the user are HTTP requests. User model updates are carried out by taking only into account the information extracted from client requests. Platform and user interface characteristics are captured in the client by means of JavaScript code that the system inserts in the generated HTML pages, and the information is returned to the server as part of the HTTP request when the user clicks on links and buttons. This assumption greatly simplifies the system architecture and the integration with external tools and modules. In exchange, it means that the system is not explicitly aware of user activity between two requests, and presentation is not updated in this interval either. A finer-grained approach could be supported by generating Java user interface components (applets) that interact with the user, and communicate directly with the server to query and update the domain and user models.

4. Conclusion

We have presented a flexible system for the dynamic generation of adaptive hypermedia presentations in terms of custom domain knowledge representations. Our approach allows the specification of presentation independently from the elaboration of contents, enhancing presentation consistency and content reuse, and reducing the development cost. The presentation language is kept simple by making a few assumptions about how ontologies are defined, and by including into the system domain knowledge about hypermedia presentation of instructional resources. To improve usability, we are currently developing a graphical authoring tool for presentation model customization by example, using similar techniques to [Castells 1999].

In most cases, our presentation system will not work alone. After a topic ontology is built, a runtime system is needed to set up and update topic networks, but this is a separate concern. Because the dynamic generation of presentation is a separated mechanism from the user and domain model update mechanisms, our presentation system can be used with different courseware support tools.

We are currently testing our system with courses on Graph Algorithms, Object Oriented Programming, Art History and Geography. In addition to extending existing courses and developing new ones, our future plans for the validation of the proposed techniques include the integration of our system with existing adaptive hypermedia support systems like [Carro 1999]. We are also planning to experiment our approach in other domains like information systems, help systems, and semantic web search.

5. Acknowledgements

The work reported in this paper is being partially supported by the Spanish Interdepartmental Commission of Science and Technology (CICYT), project number TEL1999-0181.

References

- Brusilovsky, P. (1998). Methods and Techniques of Adaptive Hypermedia. In Brusilovsky, P., Kobsa, A., Vassileva, J. (eds.), *Adaptive Hypertext and Hypermedia*. Kluwer Academic Publishers, pp. 1-43.
- Carro, R. M., Pulido, E., Rodríguez, P. (1999). Dynamic generation of adaptive Internet-based courses. *Journal of Network and Computer Applications*, 22, pp. 249-257.
- Castells, P., Szekely, P. (1999). Presentation Models by Example. In Duke, D.J., Puerta A. (eds.), *Design, Specification and Verification of Interactive Systems '99*. Springer-Verlag, Viena, pp. 100-116.
- De Bra, P. (1999) Design Issues in Adaptive Hypermedia Application Development. Proceedings of the 2nd *Workshop on Adaptive Systems and User Modeling on the World Wide Web*. Toronto and Banff, Canada, pp. 29-39.
- Fensel, D., Angele, J., Decker S., Erdmann M., Schnurr, H. P., Staab, S., Studer, R., and Witt, A. (1999). On2broker: Semantic-based access to information sources at the WWW. Proceedings of the *World Conference on the WWW and Internet (WebNet 99)*. Honolulu, Hawaii, pp. 366-371.
- Gruber, T. R. (1993). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In Guarino, N. and Poli, R. (eds.), *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Press, Boston.
- Macías, J. A., Castells, P. (2001). A Generic Presentation Modeling System for Adaptive Web-based Instructional Applications. Proceedings of *ACM Conference on Human Factors in Computing Systems (CHI 2001), Extended Abstracts*. Seattle, Washington.
- Murray, T. (1998). Authoring Knowledge Based Tutors: Tools for Content, Instructional Strategy, Student Model, and Interface Design. *Journal of the Learning Sciences*, 7 (1), pp. 5-64.
- Sun Microsystems, Inc. (2001). JavaServer Pages™ 1.2 Draft Specification. Available in the World Wide Web from: <http://java.sun.com/products/jsp>.
- Vassileva, J. (1997). Dynamic Course Generation on the WWW. Proceedings of 8th *World Conference on Artificial Intelligence in Education (AIED 97)*. Kobe, Japan, pp. 498-505.