

# An Authoring Tool for Building Adaptive Learning Guidance Systems on the Web

José A. Macías and Pablo Castells

E.T.S. Informática, Universidad Autónoma de Madrid. Campus de Cantoblanco,  
28049 Madrid, Spain.  
{j.macias,pablo.castells}@uam.es

**Abstract.** In the field of guided learning on the Internet we present, in this paper, an interactive tool for designing intelligent tutoring systems on the web. Our tool makes easier the creation of an ontology describing the content model for a given course. Such ontology contains information about classes and instances, reflecting the structure and components for the later creation of an adaptive course, using our web-based runtime course manager system. Our authoring tool generates XML code to improve course understanding as well as transportability and processing by our runtime system, which means that the generated code will reflect, in an easier way, the course structure and contents, being readable for most of users and course designers.

## 1 Introduction

The rapid development of the Internet in the last decade has given rise to new research in web-based educational technology for the creation of Intelligent Tutoring Systems (ITS) that support user adaptation and guidance. ITS's are computer-based instructional systems that have separate data bases, or knowledge bases, for instructional content (specifying what to teach), and for teaching strategies (specifying how to teach), and attempt to use inferences about a student's mastery of topics to dynamically adapt instruction [9]. Most of the efforts in this field are focused on providing intelligence on the web by means of Adaptive Hypermedia Systems (AHS). An Adaptive Hypermedia System or engine may change the content and presentation of hypermedia nodes and may alter the link structure or annotate links, based on a user model [5].

On the other hand, creating and manipulating web applications is, in general terms, an awkward task for non-expert users. In fact, nowadays there is a growing need for ITS authoring tools [10]. However few approaches exist for evaluation and generalisation because they are very difficult and expensive to build. The development of these tools is always driven by pragmatics and usability issues. In this direction, some authoring tools try to make easier the way of coding and processing course and educational information on the web, using an ontology for structuring contents and presentation. An ontology is an explicit specification of a shared conceptualisation. Its usefulness for information presentation, information integration and system development has been demonstrated recently [12].

The main goal of our work is to provide an authoring tool, called PERSEUS, used to define an ontological representation of course domain and model. Such tool allows the designer to create adaptive educational applications based on PEGASUS (Presentation modelling Environment for Generic Adaptive hypermedia Support Systems), our generic runtime system for developing adaptive hypermedia presentations in the Internet. The objective of PEGASUS is to provide course designers with a simple specification paradigm for defining non-trivial aspects of adaptive presentation independently from contents [6,7]. PERSEUS provides an object-oriented user interface for PEGASUS, improving generality and usability for building courses in different domains.

## **2 Related Work**

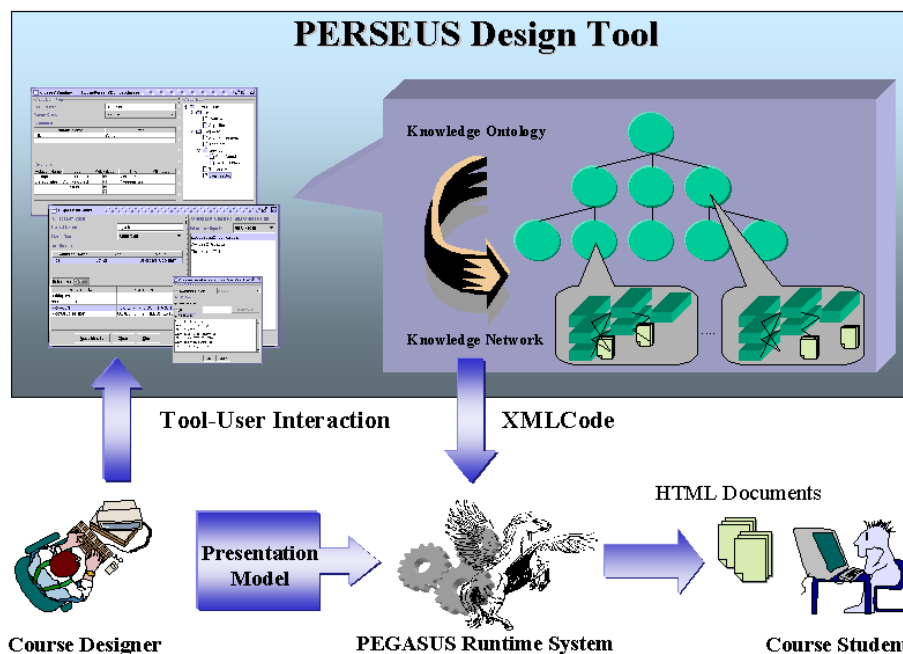
The development of web-based strategies for creating adaptive software has yielded considerable advancements in the adaptive hypermedia context over the last few years, with systems like DCG [14,15], ELM-ART [16], Interbook [1] and TANGOW [2]. ELM-ART and DCG use an explicit representation of domain concepts, interrelated in a prerequisite graph. DCG includes a planner that guides the student along a path to reach goal concepts starting from already-known concepts. ELM-ART uses a sophisticated system to estimate the knowledge acquired by the user in relation to a concept map of the course, according to which the system dynamically proposes the student a path to follow at each moment. While in ELM-ART and DCG the structure of courses is fixed, being the student itinerary what varies, TANGOW generates the course structure at runtime. TANGOW models student activity in the form of a hierarchy of tasks that represent didactic units that the student can perform. ATLAS [8] allows the fully interactive construction of courses that adapt automatically to the student's characteristics and her/his behaviour while taking the course. The designer interacts with the tool by using an intuitive visual language based on the direct manipulation of elements involved in the course. The tool takes care of the transition between a teacher's understanding of the course and the representation model of the underlying system.

The systems mentioned above lack an authoring tool that provides design support, except TANGOW, which uses ATLAS as an authoring tool for course design. ATLAS works over a fixed ontology (the TANGOW model), while PERSEUS, as a continuation of our previous work in ATLAS, allows the designer to build his/her own ontologies. In PERSEUS, course construction is achieved by defining a content model, by means of an intuitive user interface that allows the course designer to build general objects and class ontologies as basic tools for the design of the educational system.

Some systems, like KA2 [12], and Eon ITS [11], are focused on ontology engineering. KA2 has been conceived for semantic knowledge retrieval from the web, building on knowledge created in the knowledge-acquisition community. To structure knowledge, an ontology has been built in an international collaboration of researchers. The ontology constitutes the basis to annotate WWW documents of the knowledge acquisition community in order to enable intelligent access to these documents and to

infer implicit knowledge from explicitly stated facts and rules from the ontology. the EON ITS uses an ontology to define the types of topics and topic links allowed in a semantic net representation of the tutor's knowledge called the "Topic Network". Here, the ontologies are not specific to the domain, but appropriate for a class of domains. The ontologies specify a number of other things, such as topic properties (e.g. difficulty, importance), and allowed values in the student model. Knowledge is structured in EON using several mechanisms. The first is the hierarchy of basic objects: Lessons, Topics, Topic Levels, and Presentations. The second method consists of allowing arbitrary classifications of Topics and Topic links in the topic network. Given the right ontology, all the hierarchies, lattices and networks can be represented with topic networks. The third mechanism is the Topic Levels themselves. In PERSEUS, the underlying knowledge presentation is similar in many respects to EON. Our tool has only a few fixed classes previously defined. This way, each designer completes or builds his/her own class hierarchy depending on the content model of the course, or presentation, to be designed.

### 3 PERSEUS as a Design Tool



**Fig. 1.** A general view of our adaptive educational system, where PERSEUS is integrated as an authoring tool for developing course contents. Both the domain model and the presentation model will be provided to PEGASUS for generating the appropriate course feedback to the final user

PERSEUS (Presentation ontology builder for cuStom lEarning sUpport Systems) is an interactive tool for designing web-based adaptive courses. The PERSEUS interface allows designers to model courses by defining and creating an ontology of objects that are used to build adaptive presentations for the educational context (Figure 1).

This tool can generate XML code from ontologies created by the designer in the PERSEUS environment. Such XML code will be processed by PEGASUS, our runtime management system used to execute the right sequence of steps in order for any given course to be presented to the final user.

The steps needed for creating a content model are, to begin with, the creation of a class hierarchy, and, in the second place, the creation of objects that are instances of the previously defined classes. To achieve this, we just have to provide the right values for attributes and relations used to specify a certain course [6,7].

### 3.1 Creating a Knowledge Ontology

We can create any class hierarchy by just opening the class edition windows. PERSEUS provides a few predefined classes: *DomainObject*, *Topic*, *Fragment* and *AtomicFragment*. New classes can be defined by providing a class name, and the parent class from which the relevant attributes and relations are inherited. This way, we can build a whole hierarchy for a given domain. For each class, we can create class attributes, by just giving a name and an attribute type (string, number, or boolean). Later, a presentation model can be defined by associating a presentation to each class of the hierarchy.

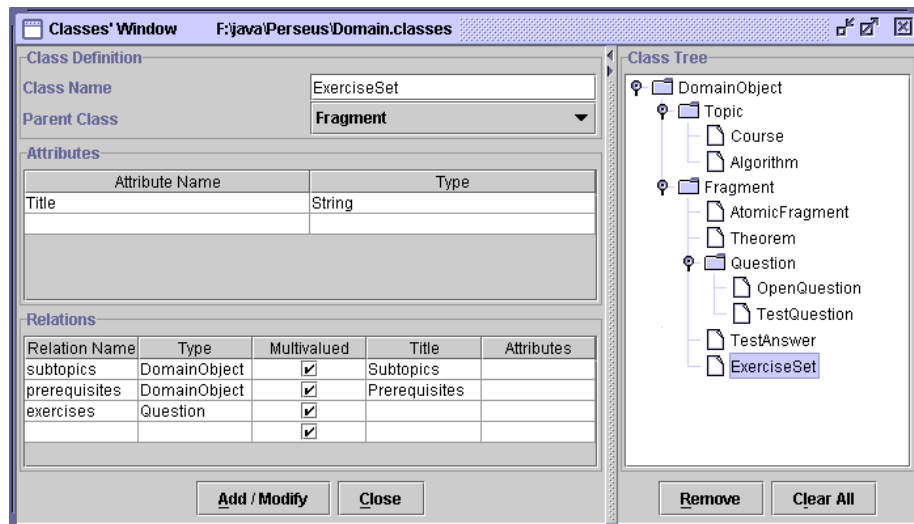


Fig. 2. Class ontology for creating a lesson about Dijkstra's Algorithm

In order to relate a class to others, we can create class relations. A class relation has a name, a given type (corresponding to the related class type), a relation title, and a boolean value to indicate whether the relation is multi-valued or not. In addition,

relations can have attributes that are defined interactively by clicking on the appropriate table cell (see Figure 2) and filling-in text fields in a pop-up dialog, where we specify the attribute name, type (string, number, or boolean), and a default value for each attribute.

The result of creating classes is a personalised tree-view of the hierarchy (right panel in Figure 2), where we can navigate by clicking on every node and deleting, if we want, the selected information. We show on Figure 2 a real example of a class ontology for creating a lesson about Dijkstra's Algorithm. In this Figure, we can see how some kinds of classes: *Algorithm*, *Course*, *Question*, and so forth, have been defined, starting from the default hierarchy mentioned previously. This ontology will be used for instantiating a course about graph theory, as we describe next.

### 3.2 Creating a Knowledge Network

Once a class hierarchy has been defined, specific objects or class instances can be created. The interface to achieve this task is shown in Figure 3, where the user can create objects from a specific class hierarchy.

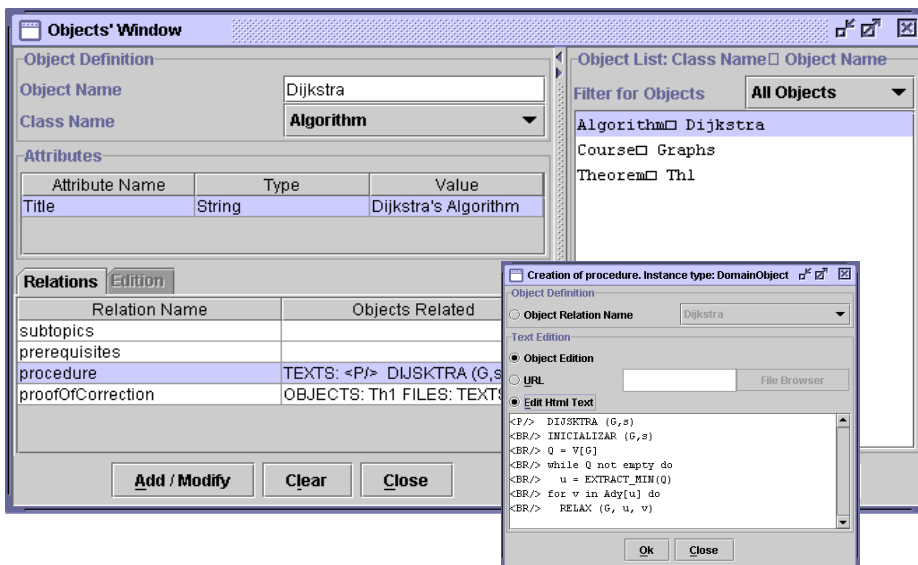


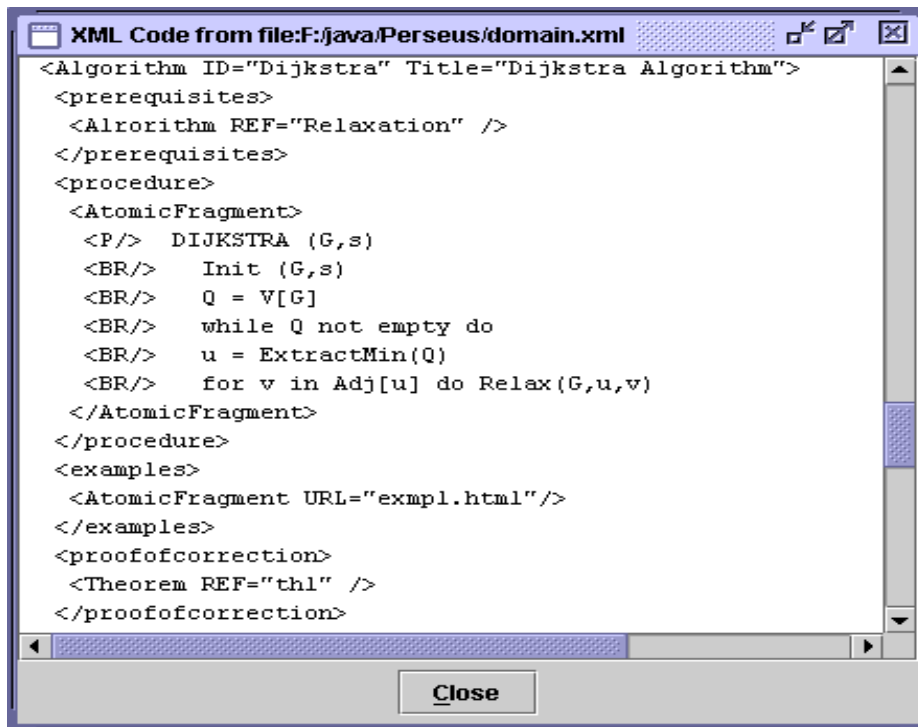
Fig. 3. User interface for the creation of class instances

In Figure 3, one can see an example of three instantiated objects, using the class hierarchy previously mentioned. *Dijkstra*, *Graphs*, and *th1*, are instances of *Algorithm*, *Course*, and *Theorem*, respectively, that will be used for creating a course about graph algorithms. As showed, each object has its own relations, based on the previously created class hierarchy. If we look at the *procedure* relation defined for the *Dijkstra* object (from the *Algorithm* class), we can see that an atomic fragment has been defined (right window of Figure 3), containing the algorithm pseudocode written

in HTML. This way, we can create objects and relate them to each other in order to build different structures for a given lesson or a complete course.

### 3.3 Generating XML Code

After creating class instances and relations between them, the next step is to generate XML code from the complete design. By clicking on the *Generate XML Code* menu command, PERSEUS automatically generates and saves into a file the XML information about the design. In Figure 4 we can see the XML code generated for the previously created Dijkstra's Algorithm object.



```
<Algorithm ID="Dijkstra" Title="Dijkstra Algorithm">
  <prerequisites>
    <Algorithm REF="Relaxation" />
  </prerequisites>
  <procedure>
    <AtomicFragment>
      <P/> DIJKSTRA (G,s)
      <BR/>   Init (G,s)
      <BR/>   Q = V[G]
      <BR/>   while Q not empty do
      <BR/>     u = ExtractMin(Q)
      <BR/>     for v in Adj[u] do Relax(G,u,v)
    </AtomicFragment>
  </procedure>
  <examples>
    <AtomicFragment URL="exmpl.html"/>
  </examples>
  <proofofcorrection>
    <Theorem REF="th1" />
  </proofofcorrection>
</Algorithm>
```

Close

Fig. 4. Window showing the generated XML code for the *Dijkstra's Algorithm* course object

Since the design information is directly generated in XML, the course information is highly portable. In particular, this makes it easy for any kind of user to read and understand, in a very simple way, the objects and the relations between them.

### 3.4 Post-processing of XML Code in PEGASUS

Finally, PEGASUS will create dynamically all course pages after reading the XML code from PERSEUS. This will be the last step for completing the final presentation.

PEGASUS allows associating a presentation model to ontology classes. The PEGASUS presentation model consists of presentation templates and presentation rules. Presentation templates define what parts (attributes and relations) of a knowledge item must be included in its presentation and in what order, their visual appearance and layout. Presentation rules are responsible for generating adaptive presentation constructs, involving relations between domain objects from very succinct high-level descriptions given in templates.

PEGASUS templates are defined by using an extension of HTML based on JavaServer Pages<sup>TM</sup> (JSP) [13], that allows inserting control statements (between `<%` and `%>`) and Java expressions (between `<%=` and `%>`) in the HTML code. In these templates, the designer can use all the presentation constructs of the HTML language (lists, tables, frames, links, forms, etc.), and insert, using very simple Java expressions, the domain items to be presented. For instance, a very simple template for class *Algorithm* could be as follows:

```
<h2> <%= title %> </h2>

<h3> Previous concepts </h3>

<%= prerequisites %>

<h3> Procedure </h3>

<%= procedure %>

<h3> Examples </h3>

<%= examples %>

<h3> Proof of Correction </h3>

<%= correction %>
```

In these templates the presentation author only needs to refer to attributes and relations of the presented class (shown in bold in the example). The presentation system takes care internally of aspects like automatically handling lists (multivalued relations like the examples of an algorithm), or recursively applying templates to referenced objects according to their class (e.g. the proof-of-correction *Theorem*'s of an algorithm). The resulting page for Dijkstra's algorithm with this presentation template can be seen in Figure 5, where HTML elements surrounding the algorithm presentation (frame structure with contextual index on the left and *Previous / Next* buttons at the bottom) come from the presentation template for the root class *KnowledgeUnit*.

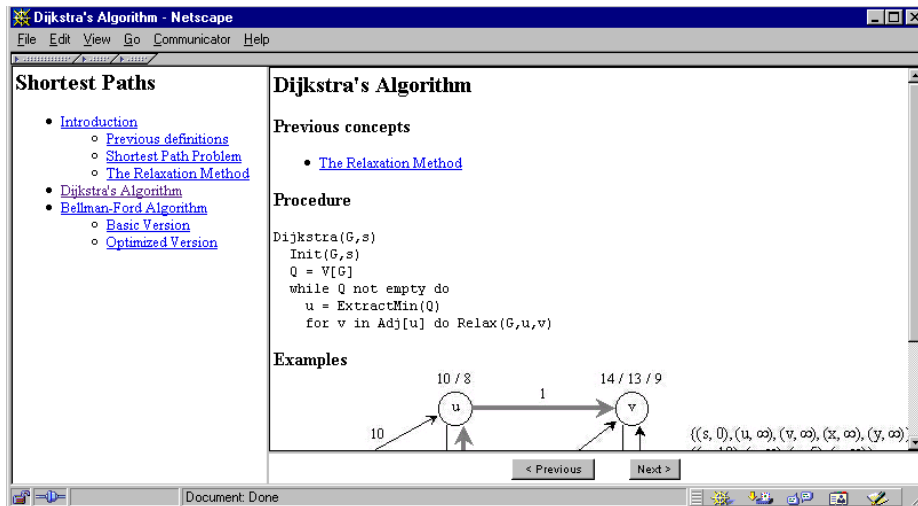


Fig. 5. Generated web page for a topic of type *Algorithm*

The template definition language supports the introduction of adaptive elements by using conditionals. For instance, in the preceding example, the presented information could be conditioned to the student's level of expertise, including all available examples when the student is a beginner, and a single example for more advanced students, showing the proof of correction only if it is relevant and not too difficult for the student:

```
<% if (user.expertise < 0.5) { %> <%= examples %> <% }
%>

<% else { %> <%= examples.upto(1) %> <% } %>

<% if (correction.relevant && correction.difficulty <
user.expertise) { %>

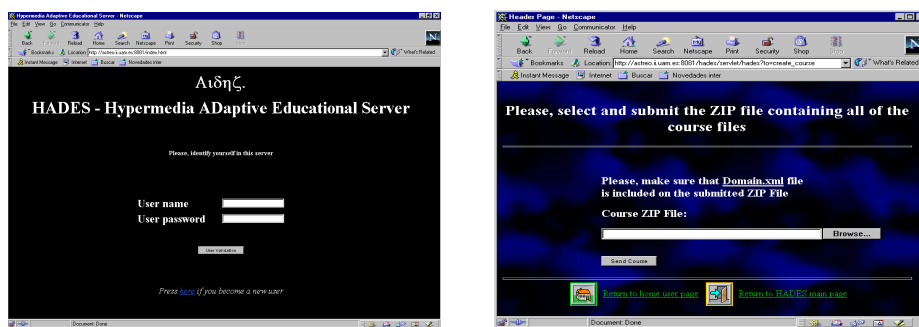
<h3> Proof of Correction </h3> <%= correction %> <% }
%>
```

The expression language for templates includes other facilities that allow, for instance, cutting down, filtering or sorting lists according to an arbitrary comparison function, generating trees and linked lists by traversing a relation, or forcing the generation of hypermedia links. The basic template language allows the specification of a wide set of non-trivial presentations by using a very simple syntax. However the designer can write arbitrarily complex Java code inside the templates themselves.

## 4 Putting all Together into HADES

As we described previously, the main goal of our work is to build a complete adaptive system in which any given course can be presented to the final user. Up to now we have described different parts of the technology used to build adaptive presentations, however a certain mechanism is required to integrate the PERSEUS design tool and the PEGASUS runtime system for creating and managing available courses.

For this purpose, we have developed HADES (Hypermedia ADaptive Educational Server), a main WEB portal providing users with an easy and fast WEB interface in order for any given course to be created or presented to the final user (see Figure 6).



**Fig. 6.** Two snapshots of HADES showing the login page (left) and the course creation page (right)

Users are registered into HADES by completing an initial form. Once the user is registered and logged to HADES, the system automatically captures the main characteristics of the user platform (e.g. language, screen resolution, navigator type and version, and so on). Moreover, each user can have one or more specific roles in the system (student, course designer, or administrator). HADES stores and reads all this information from a distributed data base management system, and it updates the user model in response to actions performed by the user in the system. HADES also has specific administration options for advanced users in order to manage users, roles and general system configuration.

To add a new course, the designer has to create a ZIP file, containing all the files required for the course construction (the XML domain file built with PERSEUS as described in previous sections, HTML fragments, JPG images, presentation templates and rules, etc.). Then the system will store and organise automatically all courseware elements, updating the HADES database of available courses for later delivery (see Figure 7).

HADES integrates PEGASUS as a runtime system for running all courses and presentations requested by any connected user. An overview of the global architecture of the system can be seen in Figure 7, where we can see PERSEUS, PEGASUS and the HADES portal resulting in a complete system for the generation and support of WEB presentations.

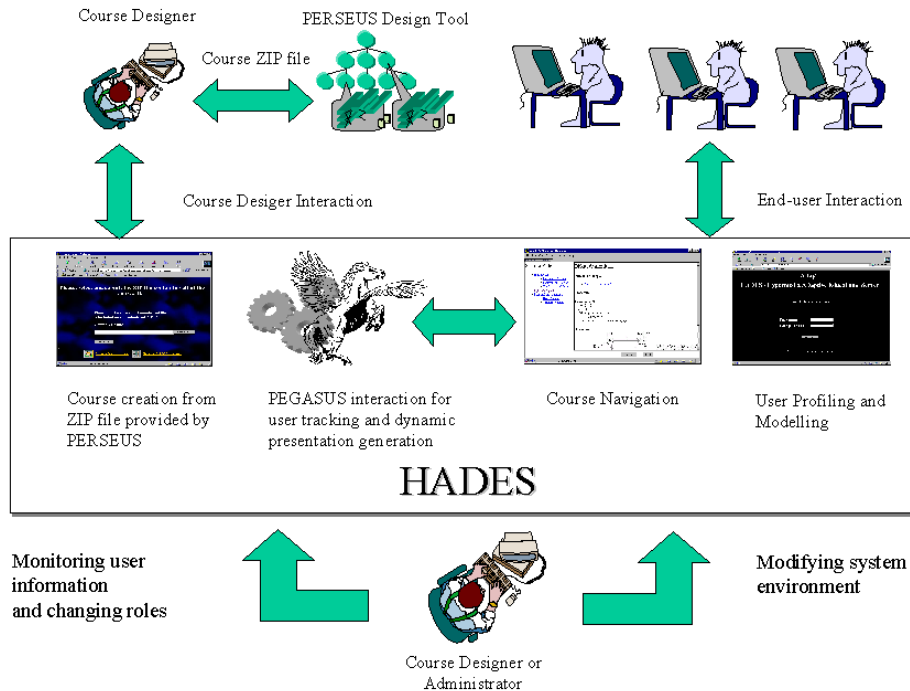


Fig. 7. Overall system architecture

## 5 Conclusions

In the context of web-based adaptive learning, PERSEUS, an interactive authoring tool for building adaptive tutoring systems, has been introduced. Our tool allows the course designer to model course information under an object-oriented paradigm, creating an appropriated ontology to define courses. Course information is translated into XML code, which makes easier the course post-processing and portability to other runtime systems like PEGASUS, our adaptive course management system. PEGASUS generates the course presentation from a presentation model defined using templates coded using JSP technology.

In addition to a tool like PERSEUS to build domain models for representing course contents, another kind of tool for creating PEGASUS presentation models is needed. The main idea of a presentation design tool is to be able, for course designers, to specify the way that the course information generated by PERSEUS will be showed to the final user. In this direction, we are currently working on a tool that infers changes to how course pages are generated, by allowing users to edit system-supplied pages directly with a basic tool like Netscape Composer. This means making it possible to infer changes, in a automatic way, on course presentation. Since such tool operates as an automatic inference system, it will get information from user actions, a modified course page in this case, and will produce adaptive changes on saved course

information. The tool will be based on the programming by example paradigm [3,4] and will be integrated into HADES.

PERSEUS, PEGASUS and HADES, are being developed in Java™ (JDK 1.3), using XLM/JDOM. PEGASUS and HADES use JavaServer Pages™ [13] for dynamic page generation. A version of PERSEUS is available from <http://astreo.ii.uam.es/~atlas/perseus/perseus.html>. HADES can be accessed at <http://astreo.ii.uam.es:8081>.

## 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

1. Brusilovsky, P., Eklund, J., Schwarz, E.: Web-based Education for all: a Tool for the Development of Adaptive Courseware. *Computer Networks and ISDN Systems*, 30, pp. 1-7, 1998.
2. Carro, R.M., Pulido, E., Rodríguez, P.: Dynamic generation of adaptive Internet-based courses. *Journal of Network and Computer Applications*, v. 22, pp. 249-257, 1999.
- 3.. Castells, P. and Szekely, P.: Presentation Models by Example. In *Design, Specification and Verification of Interactive Systems '99*, D.J. Duke and A. Puerta (eds.), pp. 100-116. Springer-Verlag, Viena 1999.
4. Cypher, A. (ed.): *Watch What I Do: Programming by Demonstration*. The MIT Press, 1993.
- 5.. De Bra, Paul.: *Design Issues in Adaptive Web-Site Development*. Proceedings of the Second Workshop on Adaptive Systems and User Modeling on the World Wide Web. pp. 29-39 Toronto 1999.
6. Macías, J.A. and Castells, P.: Adaptive Hypermedia Presentation Modeling for Domain Ontologies. To appear in *Proceedings of 10th International Conference on Human-Computer Interaction (HCI '2001)*. New Orleans (Louisiana), August 2001.
7. Macías, J.A. and Castells, P.: A Generic Presentation Modeling System for Adaptive Web-based Instructional Applications. To appear in *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI'2001)*, Extended Abstracts. Seattle (Washington), April 2001.
8. Macías, J.A. and Castells, P.: *Interactive Design of Adaptive Courses*. 2º Simposio Internacional de Informática Educativa (SIIIE'2000). Puertollano (Ciudad Real), 2000.
9. Murray, T. : *Authoring Intelligent Tutoring Systems: An analysis of the state of the art*. *International Journal of Artificial Intelligence in Education*, Vol. 10, pp. 89-129, 1999.
10. Murray, T.: *Authoring Knowledge Based Tutors: Tools for Content, Instructional Strategy, Student Model, and Interface Design*. *Journal of the Learning Sciences*, Vol. 7, No. 1, pp. 5-64, 1998.
11. Murray, T.: *Special Purpose Ontologies and the Representation of Pedagogical Knowledge*. In *Proceedings of International Conference for the Learning Sciences (ICLS-96)*, Evanston, IL, 1996. AACE: Charlottesville, VA. 1996.
12. Staab, S. et al. *Semantic Community Web Portals*. *Proceedings of the Ninth International World Wide Web Conference*. Amsterdam, May 15-19, 2000.

13. Sun Microsystems, Inc.: Java Server Pages™ Technology. <http://java.sun.com/products/jsp>.
14. Vassileva, J.: Dynamic Courseware Generation: at the Cross Point of CAL, ITS and Authoring. Proceedings of International Conference on Computers in Education (ICCE'95). Singapore, pp. 290-297. 1995
15. Vassileva, J.: Dynamic Courseware Generation on the WWW. Proceedings 8<sup>th</sup> World Conference of the AIED Society. Kobe, pp. 498-505, Japan, 1997.
16. Weber, G. and Specht, M.: User modeling and Adaptive Navigation Support in WWW-based Tutoring Systems. Proceedings 6<sup>th</sup> International Conference on User Modeling (UM97). Sardinia, Italy, 1997.