A Design Toolkit for Hypermedia Applications

Susana Montero, Paloma Díaz, and Ignacio Aedo

Laboratorio DEI. Dpto. de Informática
Universidad Carlos III de Madrid
Avda. de la Universidad 30. 28911 Leganés, Spain
{smontero,pdp}@inf.uc3m.es aedo@ia.uc3m.es

Abstract. The development process of hypermedia applications involves a variety of users with different levels of knowledge and skills. In order to get a good communication among participants, a graphical toolkit based on a method facilitates this work. In this paper, we present a design environment, AriadneTool, that allows a designer to model a hypermedia application, to validate such a design and to generate dynamically XML + SMIL implementation templates. This environment is based upon the Ariadne Method which offers a set of integrated activities to model hypermedia applications in a systematic and iterative way.

1 Introduction

Hypermedia systems, and in particular web applications, have been extremely demanded in different areas in a very short period of time. This situation has led most developers to skip the conceptual design and directly go to the implementation, producing applications of poor quality, usability and maintainability. The majority of these implementations have been made with tools such as NetObjects' Fusion or DreamWeaver which allow an automated implementation by contents and an easy set up, but do not pay attention to intrinsic features of hypermedia systems such as sophisticated navigational structures, interactive behaviours and multimedia compositions. Several hypermedia methods and theirs CASE tools have been proposed including Autoweb [4], WebML [1] and OO-H [5]. However, these methods have the following weak points (more details can be found in [6,7]):

- **Validation and integrity rules** to test the correctness, completeness and integrity of the design.
- **Contents modeling** to organize and harmonize multimedia contents both in their temporal and spatial dimension.
- **User modeling** to model different types of application users and to apply personalization as well as security constraints.
- **Security modeling** to model which contents should be delivered to which users, who can modify or personalize items and which constraints have to be applied.
- **Evaluation stage** to collect information about the potential usability of a system to improve features and functionality of application interface.

In this paper, we present a software environment, AriadneTool, to support the hypermedia application development from conceptual modeling to dynamic generation of XML+SMIL templates. The Ariadne Method [2] is aimed at dealing with the above described problems and provides the methodological foundations for AriadneTool.

J.M. Cueva Lovelle et al. (Eds.): ICWE 2003, LNCS 2722, pp. 214–217, 2003.

2 The Ariadne Method in a Nutshell

In order to cover modeling process of a hypermedia application, Ariadne proposes the process illustrated in figure 1. The first phase, `Conceptual Design`, allows us to specify the system structure and function from an abstract point of view, for next performing a `Detailed Design` where the elements defined in the previous phase are specified in a more detailed way. Moreover, the `Evaluation` phase can be performed in parallel, what allows the method to be more iterative and generate a feedback to improve the design in the earlier phases. `Validation and Integrity Rules` are not considered a phase, but a way to guarantee the completeness and integrity of the design. For more information about the method see [2].

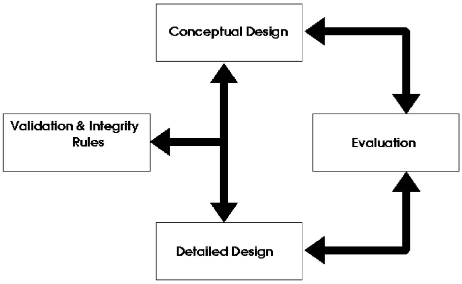


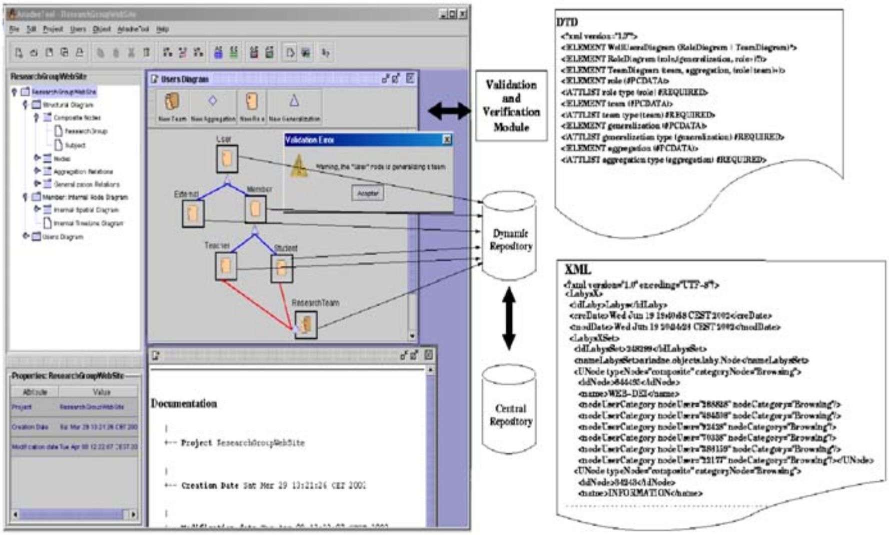
Fig. 1. The development process of Ariadne

3 AriadneTool

AriadneTool is an environment devoted to the development of hypermedia applications based upon the above design process. The main components to support such mechanisms are (see figure 2): (1) the `Front-End` that provides a perfect environment for elaborating the Ariadne Method products, which will be mentioned below, in a graphical way; (2) the `Validation and Verification Module` that holds the rules to validate and verify the completeness and correctness of the design, notifying any mistake or warning to the designer; (3) the `Dynamic Repository` that holds the components of the front-end in dynamic memory, so that access is faster; and (4) the `Central Repository` that holds the components in a persistent way.

The development environment is implemented using JDK1.4 what allows us to obtain an independent operation platform. The Validation and Verification Module is represented by DTD documents in which the rules defining well-formed design products are specified. Elements designed in the different products of the method are stored in the Central Repository by JAXB1.1 which allows an automatic two-way mapping between Java objects stored in the Dynamic Repository and final XML documents. We have built a module to translate our synchronization operations to SMIL specifications.

The main functionalities that supports AriadneTool are:



```
Documentation

|
+-- Project ResearchGroupWebSite

|
+-- Creation Date Sat Mar 29 13:21:26 CET 2003
```

```
DTD
<?xml version="1.0"?>
<ELEMENT WebUsersDiagram (RoleDiagram | TeamDiagram)*>
<ELEMENT RoleDiagram (role,(generalization, role+)?)>
<ELEMENT TeamDiagram (team, aggregation, (role|team)+)>
<ELEMENT role (#PCDATA)>
<ATTLIST role type (role) #REQUIRED>
<ELEMENT team (#PCDATA)>
<ATTLIST team type (team) #REQUIRED>
<ELEMENT generalization (#PCDATA)>
<ATTLIST generalization type (generalization) #REQUIRED>
<ELEMENT aggregation (#PCDATA)>
<ATTLIST aggregation type (aggregation) #REQUIRED>
```

```
XML
<?xml version="1.0" encoding="UTF-8"?>
<LabyrX>
 <idLaby>Labyr</idLaby>
 <creDate>Wed Jun 19 19:40:58 CEST 2002</creDate>
 <modDate>Wed Jun 19 20:04:24 CEST 2002</modDate>
 <LabyrXSet>
  <idLabyrSet>248299</idLabyrSet>
  <nameLabyrSet>ariadne.objects.laby.Node</nameLabyrSet>
  <UNode typeNode="composite" categoryNode="Browsing">
   <idNode>844495</idNode>
   <name>WEB-DEI</name>
   <nodeUserCategory nodeUser="288828" nodeCategory="Browsing"/>
   <nodeUserCategory nodeUser="494598" nodeCategory="Browsing"/>
   <nodeUserCategory nodeUser="92428" nodeCategory="Browsing"/>
   <nodeUserCategory nodeUser="70338" nodeCategory="Browsing"/>
   <nodeUserCategory nodeUser="288159" nodeCategory="Browsing"/>
   <nodeUserCategory nodeUser="22177" nodeCategory="Browsing"/></UNode>
  <UNode typeNode="composite" categoryNode="Browsing">
   <idNode>34243</idNode>
   <name>INFORMATION</name>
```

Fig. 2. The AriadneTool architecture

- The *Structural Diagram*. It depicts the logical structure of the application by means of nodes (simple or composite) and connect them through two abstraction mechanisms: generalization and aggregation.
- The *Navigation Diagram*. It specifies the navigation structure by means of tagged links and navigation tools.
- The *Attributes Catalogue*. It is a repository where properties of elements are defined by a name/default-value pair.
- The *Events Catalogue*. It is a repository where actions that take place at runtime when a condition is fulfilled can be defined and reused in different scenarios.
- The *Internal Diagram*. It collects multimedia contents, attributes and events, as well as anchors of a node. The Internal Diagram is made up of a Spatial and Timeline Diagram.

  The *Spatial Diagram* lets specify logically the appearance of a node by placing contents and space-based relationships among them. The *Timeline Diagram* allows us to represent how the node evolves throughout a time interval. Time-based relationships can be defined among contents to specify delays in the beginning/end of the target content.
- The *Users Diagram*. It represents the kinds of users in a hierarchical way using two abstractions: roles and teams. A role is a responsibility or job function but no an individual user meanwhile a team represents a group of roles which join together to work.
- The *Categorization Catalogue*. It holds the security category assigned to each node or content to determine the most permissive operation that can be performed over it by a user. These categories are no access, browsing, personalizing and editing.

- The *Access Table*. It allows us to define the security policy of the system assigning a manipulation category to each role for each node and content. Thus the actions that users can perform are controlled. More details on the security model for hypermedia applications applied in this method can be found in [3].
- The validation and verification of the above products to check completeness and consistency of the model.
- The automatic documentation generation about the made design.
- The assignment of data to multimedia contents in order to generate dynamically XML+SMIL templates from which, for example using XSLT stylesheets, the web application can be set up.

4 Conclusions

In this paper, we present an environment, AriadneTool, to support the design and development of hypermedia applications following the Ariadne Method, which pays special attention to products to model users structures, interactive behaviours, multimedia compositions and security policies. Moreover, we have evaluated AriadneTool by collecting feedback from the students in a course on Hypermedia Design. This feedback has been very positive, and a large majority believed that the tool supported well the method tasks and made much easier to use the Ariadne Method.

Acknowledgements. We'd like to thank Jose Ángel Cruz for his cooperation in the development of AriadneTool. This toolkit is part of the Ariadne project funded by "Dirección General de Investigación del Ministerio de Ciencia y Tecnología" (TIC2000-0402)".

References

1. S. Ceri, P. Fraternali, and A. Bongio. Web modeling language (WebML): a modeling language for designing web sites. *WWW9 / Computer Networks*, 33(1-6):137–157, 2000.
2. P. Díaz, I. Aedo, and S. Montero. Ariadne, a development method for hypermedia. In *proceedings of Dexa 2001*, volume 2113 of *LNCS*, pages 764–774, 2001.
3. P. Díaz, I. Aedo, and F. Panetsos. Modelling security policies in hypermedia and web-based applications. In *Web Engineering: Managing diversity and complexity of web application development*, pages 90–104, 2001.
4. P. Fraternali and P. Paolini. Model-driven development of web applications: the autoweb system. *ACM Transactions on Office Information Systems*, 18(4):323–282, 2000.
5. J. Gómez, C. Cachero, and O. Pastor. Conceptual modeling of device-independent web applications. *IEEE MultiMedia*, 8(2):26–39, 2001.
6. S. Montero, P. Díaz, and I. Aedo. Requirements for hypermedia development methods: A survey of outstanding method. In *Proc. of Advanced Information Systems Engineering, 14th International Conference, CAiSE*, pages 747–751, 2002.
7. S. Montero, P. Díaz, and I. Aedo. A framework for the analysis and comparison of hypermedia design methods. In *Proc. of The IASTED International Conference on Software Engineering (SE'2003)*, 2003.