Modeling Hypermedia Applications with *HyDev*

P. Pauen, J. Voss, H.-W. Six
Praktische Informatik III, FernUniversität Hagen
58084 Hagen, Germany
Email: {peter.pauen|josefvoss|hw.six}@fernuni-hagen.de
Telephone: +49-2331-987-{4647|2573|2964}
Fax: +49-2331-987-317

Abstract
This paper introduces the *HyDev* approach to a structured and systematic development of hypermedia applications. *HyDev* focuses on the early phases of the development process, i.e. analysis and design. The requirements and key aspects of the software to be built are captured with tightly coupled description models. The main emphasis of this paper lies on the models for the requirements engineering phase which, simply spoken, capture structure, content and presentation of a hypermedia application at an appropriate level of abstraction.

Keywords
Hypermedia, multimedia, authoring, software engineering, requirements engineering, modeling, model-based software development, *HyDev*

1 INTRODUCTION

In the last years importance and distribution of hypermedia applications - in the following abbreviated as HMA - have significantly increased. The range of HMAs, e.g. electronic books, multimedia learning/training software as well as product catalogs and presentations, is very inhomogeneous. Some of these primarily appear as documents, others can be more appropriately characterized as complex software systems.

There is also an increasing interest in development methods. However, traditional software development methods are mostly inappropriate - HMAs have several characteristics which distinguish them from conventional software. Probably most noticeable is the multimedial representation of the application's information and objects: In addition to text HMAs can contain graphics, pictures, and elements with a temporal dimension, like audios, videos, and animations. Secondly, HMAs are characterized by special elements and structures. For example, the flow of what is presented or happens can be determined by a kind of film-script. Typically, this can be observed in guided tours or interactive comics. In other words: HMAs can have a narrative structure. Further special elements are agents and 2D-/3D-objects. Another difference refers to user operations: Most important is navigation, i.e. is the selection of objects to be presented. Less important is typical information processing like the creation of new objects, computation, and object modification.

HMAs usually are of a considerable complexity. For example, common computer games often have a complex inner structure and audiovisual organization as well as a high degree of interaction. Therefore, specific development methods with suitable milestones and documents are principally advisable and conducive in the domain of hypermedia applications.

Established software engineering methods work well in areas like data processing, engineering or telecommunication software but can not be directly applied to multimedia applications. There are a few specialized approaches to a systematic development of HMA (see chapter 3). But these are still too immature and can not handle the complex structure of HMA at an appropriate level of abstraction. Commercial authoring tools, e.g. macromedia's Director, concentrate solely on the implementation and do not offer support for the early phases of the development process, i.e. analysis and design.

For these reasons HMA development is usually quick&dirty, resulting in low correctness, robustness, and maintainability of the end products. Consequently, practitioners like for instance Kathy Kozel (Kozel, 1996) complain about the negative consequences for the practical development work.

On the grounds of these observations we have developed *HyDev*[*], a domain-tailored approach to structured and systematic development of HMAs that explicitly takes the above mentioned characteristics into account.

The rest of this paper is organized as follows. The second chapter gives an overview over *HyDev* and the activities in the various phases of the development process. In chapter 3 we look at related work in this field. As our main contribution we subsequently introduce *HyDev*'s three requirements models in more detail. Chapter 4, 5, and 6 deals with the domain model, the instances model resp. the representation model. The paper concludes by summarizing our contributions and discussing future work to be done.

*The acronym *HyDev* is made up of the words hypermedia and development.

2 THE HYDEV APPROACH

2.1 Overview

HyDev works with several distinct models with fine-grained relationships between model elements. The various models build on one another and are the cornerstones of the development in that they capture certain aspects and decisions which are of great importance for the system to be built. Each model views the application under development from a different perspective, has a certain level of abstraction, and is intended for a specific development phase. In this respect *HyDev* is a model-based approach.

2.2 Requirements analysis

In contrast to conventional software development requirements analysis for hypermedia applications mainly deals with content and quality of presentation. Requirements resulting from the system's environment and the users' work context are much less important. Nevertheless, the structured development of an HMA should have an explicit requirements engineering phase in which the main features of the system to be built are identified and documented in a non-technical form. Mastery of the complexity succeeds primarily by abstraction, i.e. limitation to selected important aspects.

In this context it is of high importance that requirements engineering level documents do not anticipate the actual implementation. In particular, there should be no assumptions concerning the media objects. For example, it should not be necessary to know which particular videos or audios will be integrated into the final HMA. Quite important also is that an approach does not force the developer to use a specific authoring tool, nor should it restrict the developer concerning the choice between an authoring tool or a special hypermedia program library.

Roughly spoken, the aspects captured by *HyDev* requirements analysis models are structure, content and presentation of the hypermedia application. The models are called domain model, instance model and representation model, respectively. An important issue in choosing modeling concepts is to appropriately deal with the document-software dichotomy. HMAs are both: documents with individually presented objects as well as software systems which present and manipulate uniformly structured objects. In the following we briefly characterize the three models.

The need for an instances model

The navigational structure of an HMA is mainly determined by its content, i.e the objects to be presented. We have observed that in this respect instances and their relationships are at least as important as the underlying classes and their relationships. As a typical example we consider the development of a CBT* course. There one must decide in detail <u>which</u> subchapter a certain chapter has, <u>which</u> references are therein and which examples and assignments are included. So, in contrast to conventional software it is not sufficient to simply employ an ER-model or an OOA-model. Rather, the application's underlying objects have to be explicitly considered in a separate instances model. Chapter 5 contains more details about the instances model.

The need for a domain model

The consideration of objects is only conducive if corresponding classes and their relationships have been carefully modeled beforehand. Therefore like with conventional software a class model - also called domain model - is necessary. Such a class model has to be tailored to the special kinds of objects that we can find in HMAs (e.g. narrative units, agents, 2D-/3D-items). However, this kind of domain modeling, e.g. in the form of an adapted or extended variant of OOA, is entirely missing in existing approaches. The details about the domain model and its special kinds of classes can be found in chapter 4.

The need for an representation model

On the basis of a domain model and an instances model it is not appropriate to proceed directly with the implementation using concrete media objects. Obviously, domain and instances models leave a considerable degree of freedom for design decisions. We believe that it is essential to specify these aspects during the requirements engineering phase. But for two reasons it is advisable to do this at a higher level of abstraction. On the one hand the early specification of details such as position, layout, color and timing would anticipate the actual implementation and result in a unnecessary high effort of revision. On the other hand such abstraction helps mastering complexity.

Thus, simply spoken, it is specified in which way objects are presented to the user by the running application, i.e. as text, graphic, video, vrml-world or the like. Further specifications concern the navigational structure as well as user interactions. A specific model is needed for such aspects of the object representation. Another argument for this model is that logical structures between objects do not correspond directly to structures between object representations. It is very well possible that an object has several different representations. Conversely, several objects may have one common representation. Chapter 6 is devoted to the representation model.

2.3 Specification & design

During the *specification&design* phase the requirements and decisions captured with the three analysis models are concretized and refined. This way one finally obtains a complete specification of the system to be build. The decisions of this phase concern inter alia:

*CBT = Computer Based Training

- user interface objects (buttons, menus, ...)
- details of the temporal and spatial relationships between representations
- playback parameters
- media objects and playback effects
- playback channels
- details of the interaction techniques
- quality of service
- requirements on the underlying hardware of the target system

2.4 Implementation

The results of the specification&design phase serve as a starting point for the *implementation phase*. Among the activities of this phase are:

- creation of the media objects
- actual implementation and realization (e.g. with the help of an authoring tool)
- programming
- realization of effects
- adaption to the hardware of the target machine

3 RELATED WORK

In this chapter we take a look at selected other methods for structured hypermedia design and work out differences between these and *HyDev*. We will restrict ourselves to the *Relationship Management Methodology* (*RMM*) (Isakowitz et al., 1995) and the *Object-oriented Hypermedia Design Model* (*OOHDM*) (Schwabe and Rossi, 1995). Other modeling approach in the hypermedia field, like Dexter (Halasz and Schwartz, 1994) and AHM (Hardman and Bulterman, 1994), offer technology independent modeling concepts for hypermedia documents. They do not deal with development methods and rather closely stick to the document paradigm.

RMM is based on the Entity-Relationship model. The first step in the development process is a conventional ER diagram which captures the information domain of the application. This is followed by the definition of so-called slices, i.e. meaningful groups of an entity's attributes. The result of this step is an enriched ER diagram, the ER+ diagram, containing entities, relationships, and slices. Next the navigational design is developed. All navigational paths are derived from relationships between entities. They are specified in terms of entity properties and relationships. This step results in the so-called RMDM diagram, the cornerstone of RMM. The following steps comprise the conversion protocol design (each element of the RMDM diagram is transformed into an object in the target machine, for example a listbox) user-interface design (i.e. the design of screen layouts for every object of the RMDM diagram), and runtime behavior design (dealing with aspects such as link traversal, backtracking, and navigational mechanisms).

OOHDM is a model-based approach that comprises four main activities: conceptual design, navigational design, abstract interface design, and implementation. During conceptual design an object-oriented domain model with classes, relationships, attributes, and subsystems is defined. OOHDM views an HMA as a navigational view over the conceptual model. The navigational structure is defined by a schema specifying navigational classes such as nodes, links, and access structures. While nodes represent views on conceptual classes, links are derived from conceptual relationships. In the abstract interface design phase an abstract interface model is built. It captures which interface objects the user will perceive, the way in which navigational objects will appear, how navigation is activated, and synchronization aspects.

Both RMM and OOHDM use plain class models which do not take into account that HMAs can have special elements like narrative structures, spatial objects or agents. Therefore, they have to model such elements in a more complicated and less comprehensive way. Apart from that, they work only with an ER model resp. a class model. The consideration of individual objects is left out. Consequently, representations of concrete objects do not occur. We consider this a significant shortcoming. For example, we believe that many navigation connections only make sense with object representations. RMM and OOHDM appear to be most suitable for applications with uniformly structured data like, for instance, product catalogs. More complex types of HMAs, like ingenious games, will require more sophisticated specification techniques.

4 HYDEV'S DOMAIN MODEL

The idea behind *HyDev* is that an HMA is based on a collection of various objects and (potentially complex) relationships between them. In the scope of the domain analysis the corresponding classes and relationships are specified in a domain model. Since HMAs can have special elements that are usually not found in conventional software, *HyDev*'s domain model works with the following specialized classes in addition to conventional classes:

- classes of narrative structuring units: *N-classes*
- classes of objects with a spatial dimension: *S-classes*
- classes of agents: *A-classes*

These specialized classes can have attributes and operations and can be connected by associations, inheritance-, and part-of-relationships just like conventional classes. Additionally, there are numerous specialized relationships (described below in the context of according classes). By these means, *HyDev* allows a much more adequate domain modeling than a conventional OOA or ER model.

The notation of the domain model follows UML (Booch et al., 1997). Therefore, associations are notated by simple annotated lines between classes, and part-of- and inheritance-relationships by lines with rhombuses resp. arrows. The specialized relationships are marked with special symbols (see below). In order to make the different kinds of classes distinguishable, the symbols for the specialized classes have a small signet in the upper left corner: a ≈ for N-classes (symbolizing flow), a ❒ for S-classes (symbolizing a spatial cube), and a 🧍 for A-classes (symbolizing a man).

4.1 Classes for narrative units (N-classes)

In many HMAs, especially sophisticated games, one can find a *narrative structure*. For example, the interactive animated online comic Madleine's mind (Madmind, 1997) is organized in acts, episodes, scenes, and steps. A narrative structure is based on special objects that have the character of film-scripts. These objects are called *narrative units*. They are used for modeling the flow of what is presented or what happens. Narrative units structure an HMA concerning its thematic contents by grouping elements on the grounds of narration. It is possible that multiple narrative units constitute complex narrative units. They are modeled by N-classes.

Among the specialized relationships between N-classes are:

- the *sequence-relationship* by which the narrative or logical sequence of narrative units, i.e. the narrative structure is modeled. For example, in an animated multimedia comic the various scenes follow each other.
- the *simultaneity-relationship* for modeling narrative simultaneity of narrative units. For instance, such a relationship can be used to express that two parts of a story happen at the same time.
- the *prerequisite-for-relationship* that allows to specify that one narrative unit is a prerequisite for one or several other narrative units. For example, a flashback can be necessary for the comprehension of subsequent episodes.

Beyond that there is a *participate-in-relationship* that can exist between N-classes and the other kinds of classes. It aims at the objects that participate in a narrative unit. For example, a scene can take place in a certain room in which some agents appear; the room and the agents are then participants of that scene.

As an example we look at an animated multimedia comic. The comic might consist of several storylines which can happen simultaneously. Each part consists of episodes and maybe a flashback. Episodes follow each other or a flashback. A flashback can be a prerequisite for an episode. Both consist of scenes. The next picture shows the corresponding N-classes along with their attributes and the relationships. Since flashbacks and episodes are very similar the model has a common superclass for them (Block).

4.2 Classes for objects with a spatial dimension (S-classes)

HMAs often have objects that are characterized by their *spatiality*, for example rooms or 2D- or 3D-objects within rooms. These *objects with spatial dimension* are

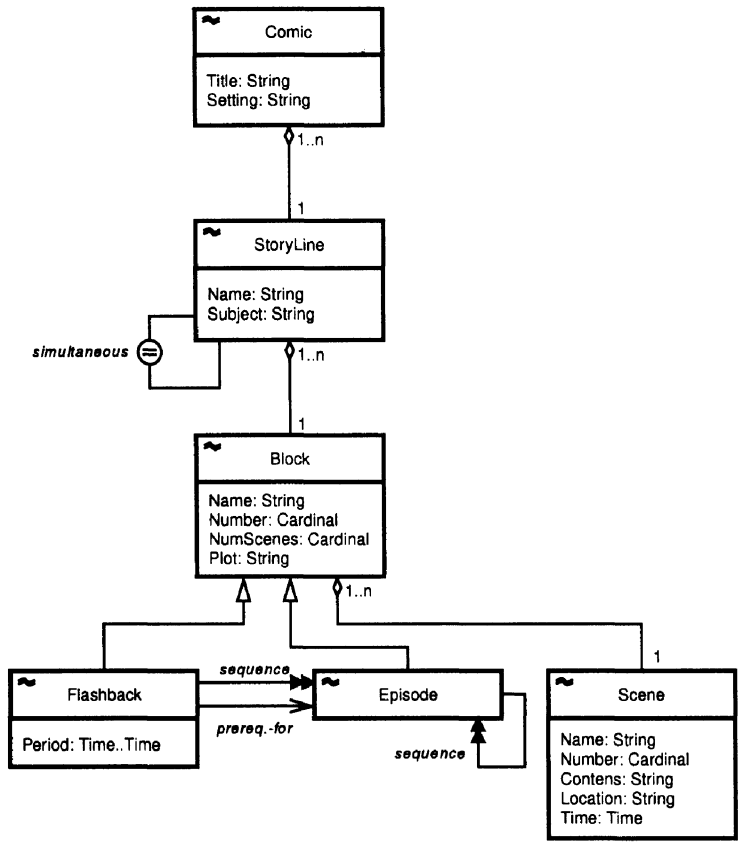


Figure 1 N-classes of a comic application.

often found in games, animated interactive comics or virtual reality software. For modeling of these objects *HyDev*'s domain model has S-classes. The ***adjacent-relationship*** or the ***contained-in-relationship*** for instance are among the many special relationships between S-classes.

Objects with a a spatial dimension can have a specific dynamic behavior. A good example are pyrotechnic articles that can be viewed in a digital product catalog for fireworks. Pyrotechnic articles are objects that have a specific behavior: They have a certain flight behavior and produce certain light and sound effects. Such aspects are modeled with the help of N-Classes or - analogous to attributes and operations - with simple behavior descriptions as part of the corresponding class definition.

In the next picture an extract of the corresponding domain model can be seen. As there are two kinds of pyrotechnic articles, PyrotechArticle has two subclasses: Rocket and FireCracker, each of which defines additional special attributes. According to the remarks above, the class definition for PyrotechArticle has three subsections for the specification of attributes, operations, and behavior. For modeling the complex orchestration of fireworks we include an N-class Orchestration. It is connected with the class Fireworks via another special relationship not mentioned yet, the ***arrangement-relationship***. Participants in such an orchestration are rockets and fire crackers. Therefore, Orchestration has a participate-in-relationship to PyrotechArticle.

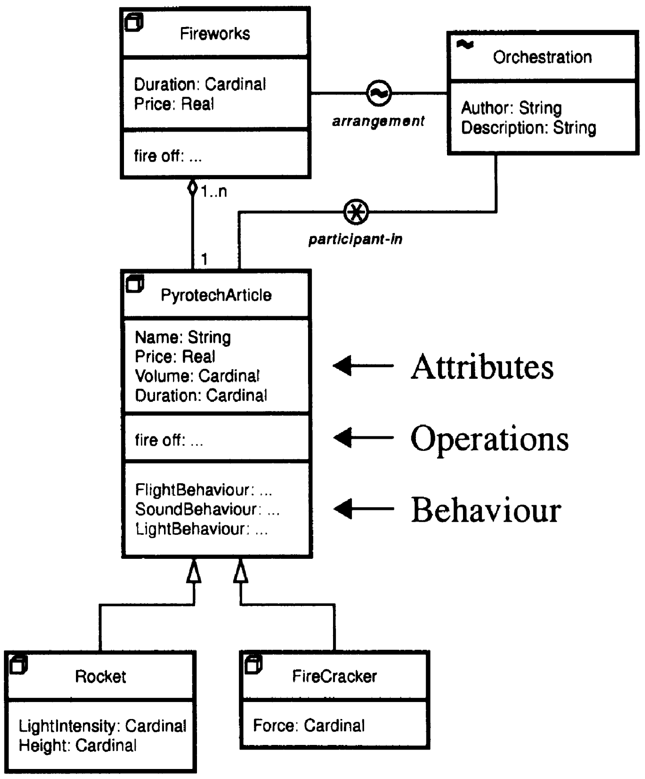


Figure 2 S-classes of product catalog for fireworks.

4.3 Classes for agents (A-classes)

Finally, HMAs - especially games and virtual reality applications - can have elements which are characterized by some kind of independence and autonomy. These elements are called agents. They are modeled with A-classes. Typical agents are characters (e.g. in adventure games) or guides trough virtual worlds. Agents never stand alone for themselves but always participate in narrative units. They also can have a certain behavior which is then modeled analogous to the behavior of objects with a spatial dimension. The behavior is the expression of their autonomy. Agents can have certain tasks, pursue an aim (within limits), react to modifications of their environment, and interact with each other.

Let's consider a typical tactical game with a hero and opponents such as helicopters and tactical groups consisting of soldiers. Obviously, these are agents that are to be modeled by A-classes. The classes Helicopter and TacticalGroup have a common superclass Opponent. Both the hero and his opponents participate in a battle which is modeled by an N-class Battle. The interaction between them is expressed by an *interaction-relationship* between the classes Hero and Opponent. The following picture shows the classes and their relationships.

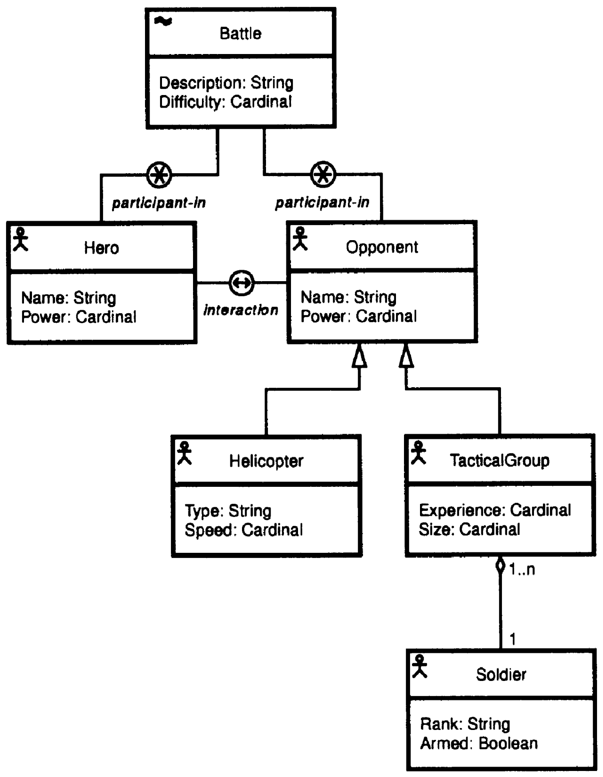


Figure 3 A-classes of a tactical game.

4.4 Example: virtual museum application

As a larger example we consider a typical virtual museum application. The museum consists of sections each of which deals with a specific theme. A section has several adjacent rooms containing the museum's exhibits. There are three kinds of exhibits: paintings, pieces of furniture, and installations (= works of art consisting of sub-objects that move in a complex way). A user can undertake tours through a section of the museum. Such a tour has a specific theme and consists of successive segments. The segments themselves are composed of steps. Tours are guided by a museum guide which walks from room to room and comments on the exhibits therein.

This application is a good example for an HMA that uses all four types of classes at the same time. Therefore, it can easily be shown how the different kinds of classes work together in a common model.

The following picture shows an extract of the domain model for the virtual museum application. To avoid cluttering, attributes, operations and behavior were omitted.

The museum, its sections and rooms as well as the exhibits are spatial objects. Consequently they are modeled with S-classes. Since pieces of furniture, paintings and installations are special exhibits, there is an inheritance-relationship between the corresponding S-classes. The fact that rooms are adjacent and contain the exhib-

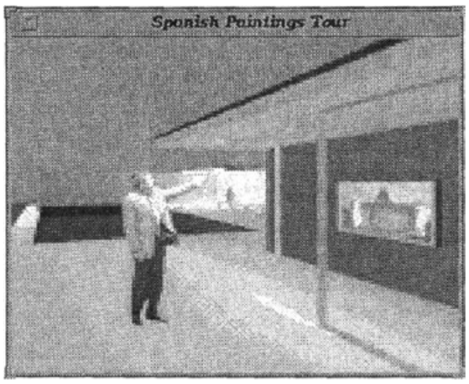


Figure 4 A guided tour through the virtual museum application (picture taken from (Gibbs and Tsichritzis, 1995)).

its is expressed by an adjacent- resp. a contains-relationship. The museum guide is a typical agent and is therefore modeled by an A-class. His complex behavior is specified with a separate N-class (CommentOnExhibit). The tour and its parts are narrative units for which the model contains N-classes. The simultaneity of the steps as part of a tour segment, and the behavior of the tour guide are taken into account by a simultaneous-relationship. Participant-in-relationships model which objects participate in the narrative units. Finally the theme of the tour is specified by a conventional class.

5 THE INSTANCES MODEL

The instances model consists of instances of the domain model's classes, and instantiated relationships. It contains a model component for every object of the running application. Each model component has a unique name and provides the name of the object's class. It is left to the developer whether he specifies the values of an object's attributes, as well as operations and behavior differing from the general specifications given in the class definition.

However, the instances model is not created by a simple and mechanical instantiation of domain model classes. For example, it is possible to aggregate objects into homogeneous or heterogeneous collections. Thus, the instances model can contain objects for which a corresponding class can not be found in the domain model.

The instances model is notated graphically like the domain model. Thus, instances are represented by nodes, and relationships by lines between them. A node's symbol has a small signet in the upper left corner indicating the object's kind of class. The lines for the different kinds of relationships are notated the same way as in the domain model, i.e. with special arrows or symbols. To avoid huge and confusing model graphs it is possible to split the instances model into several parts focussing on groups of objects that belong together.

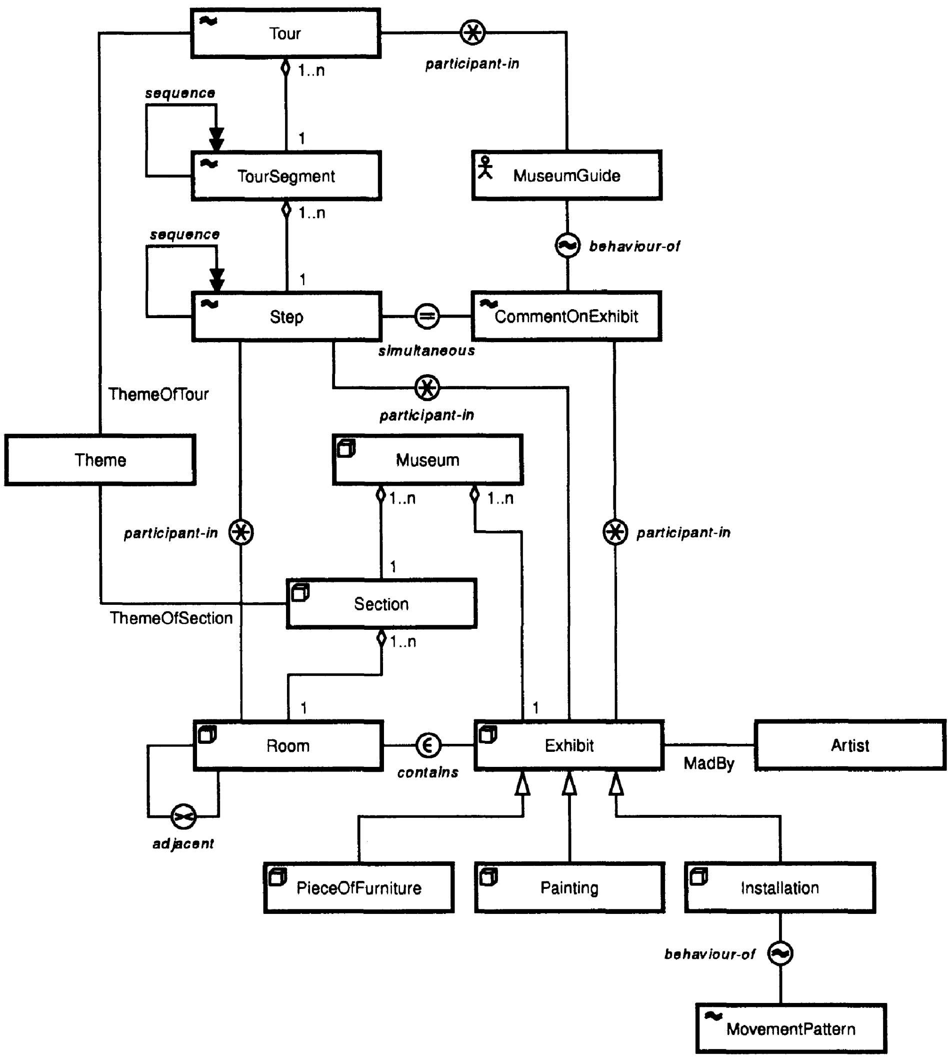


Figure 5 Domain mode for the virtual museum application.

5.1 Example: virtual museum application (continued)

Continuing the example from subchapter 4.4 we now present the instances model of the virtual museum application. The following picture shows an extract that deals with a segment of a tour through the basement of the Prado museum. This section contains among others the *pinturas negras* paintings by Goya.

Worth mentioning is the object called PintNegras. Notice that there is no class for it in the domain model. The reason is that this object is a collection of objects of the class Painting.

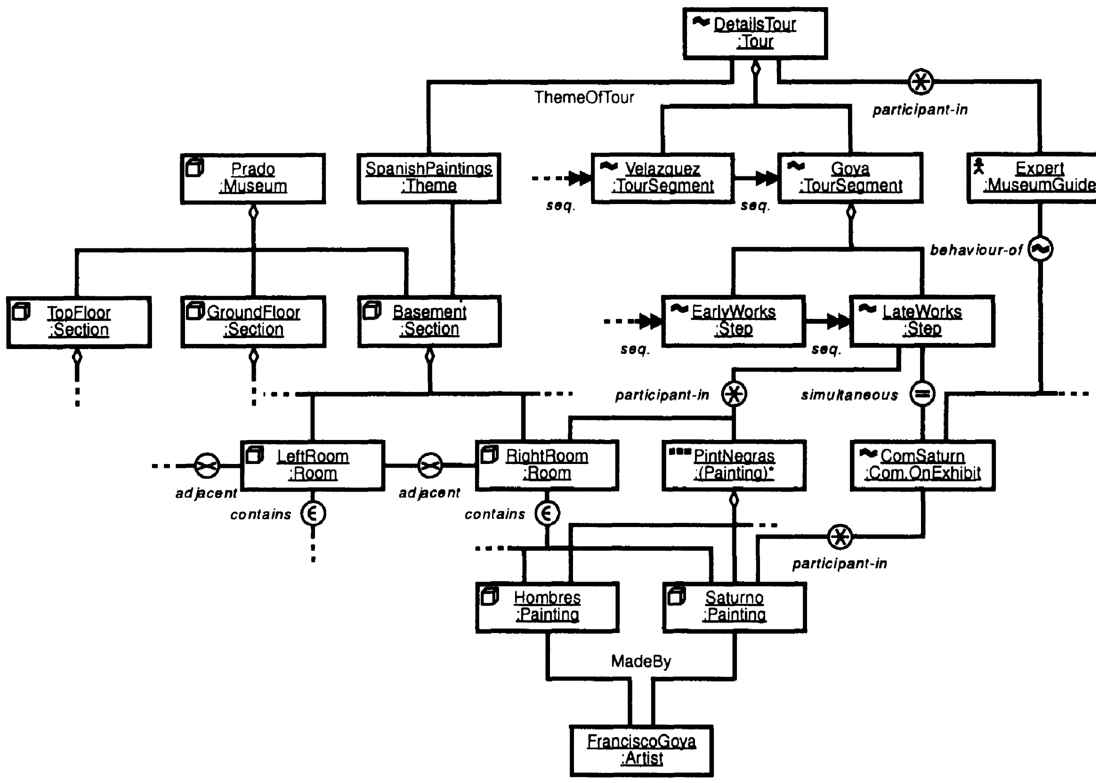


Figure 6 Instances model for the virtual museum application (extract).

6 THE REPRESENTATION MODEL

The representation model refers to the domain and instances model and captures aspects of the object representation and the user interaction. It is of vital importance that the representation model does not get overloaded with details that are secondary at this early development stage. Therefore, only the most relevant structural aspects are considered.

Its main model concept are *representations*. Primarily, a representation models which and how attributes and relationships of an object of the running application are represented to the user. To this end, the media object type (text, graphic, image, audio, video, animation, vrml-world, ...) and a list of output media (window(-part), audio channel, external device, ...) to be used for playback are specified. But a representation still abstains from details such as formats (GIF, JPEG, MPEG, ...) or even concrete names of data files.

In addition it is modeled how several representations build more complex representations. This way it is possible to establish the inner structure of the overall HMA. Besides, this helps mastering the complexity.

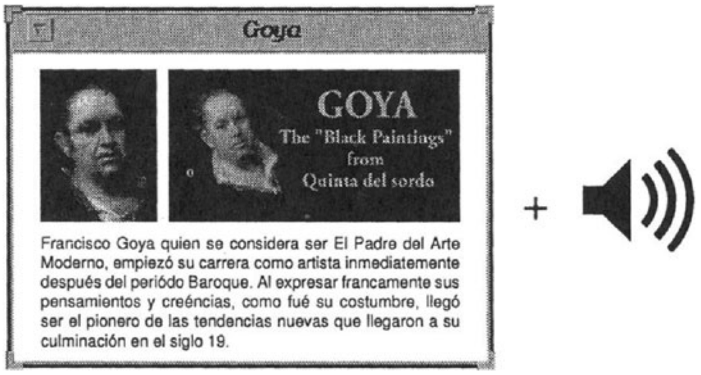
After the representations are modeled, they are interconnected via spatial-temporal relationships. These relationships specify where and/or when a representation is represented in relationship to one or more other representations. These specifications are very coarse on purpose; we just decide whether a representation is represented left, over, simultaneously, after and so forth of another representation.

We consider a navigation as a user-triggered start of playback of an object, executed at another representation. Navigation is specified like spatial-temporal relationships. Depending on whether or not a navigation leads to the termination of the playback of the representation where the navigation was executed, the corresponding line between the two participating representations has one or two arrows in the representation model.

Finally events and user commands are modeled by specifying where they have happened (e.g. in the context of a represented object), what has happened (for example, an audio has reached its end or special object appears in a video) and what the reaction is (e.g. a certain representation changes its size or location). User commands are considered as special events that are executed by the user with the help of certain interaction techniques.

6.1 Example: virtual museum application (continued)

We suppose the virtual museum application lets the user inquire detailed information about a painter by clicking on the pictures of his paintings during the tour. As a result a portrait of the painter and a short text with general information appear in a separate window. At the same time the text is played back as an audio. As soon as the user clicks on certain words within the text, the audio is stopped and a video about the painters life is played back to the right of the portrait. A short while after its end a second video about the influence of the artist follows automatically.



Francisco Goya quien se considera ser El Padre del Arte Moderno, empiezó su carrera como artista inmediatemente después del periódo Baroque. Al expresar francamente sus pensamientos y creéncias, como fué su costumbre, llegó ser el pionero de las tendencias nuevas que llegaron a su culminación en el siglo 19.

Figure 7 Window with the details about Goya.

The following picture shows an extract of the corresponding representation model. We assume, that the class Artist has the attributes Portrait, Biography, and Influence. By means of this example, it can be seen how several related representations can be bundled to a complex representation.

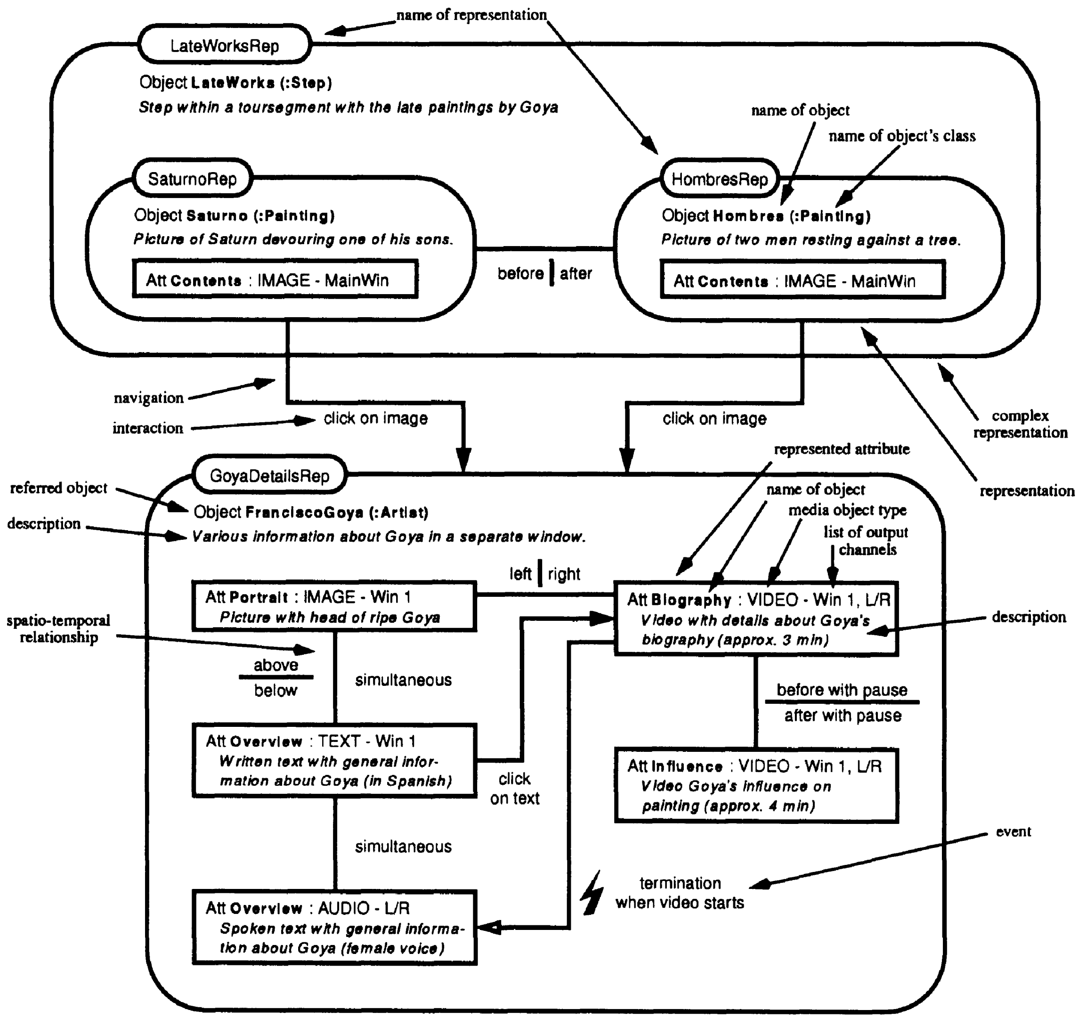

name of representation
LateWorksRep
Object LateWorks (:Step)
Step within a toursegment with the late paintings by Goya
name of object
name of object's class
SaturnoRep
Object Saturno (:Painting)
Picture of Saturn devouring one of his sons.
Att Contents : IMAGE - MainWin
before | after
HombresRep
Object Hombres (:Painting)
Picture of two men resting against a tree.
Att Contents : IMAGE - MainWin
navigation
interaction
click on image
click on image
complex representation
represented attribute
representation
referred object
description
GoyaDetailsRep
Object FranciscoGoya (:Artist)
Various information about Goya in a separate window.
name of object
media object type
list of output channels
left | right
Att Portrait : IMAGE - Win 1
Picture with head of ripe Goya
Att Biography : VIDEO - Win 1, L/R
Video with details about Goya's biography (approx. 3 min)
description
spatio-temporal relationship
above
below
simultaneous
before with pause
after with pause
Att Overview : TEXT - Win 1
Written text with general information about Goya (in Spanish)
click on text
Att Influence : VIDEO - Win 1, L/R
Video Goya's influence on painting (approx. 4 min)
simultaneous
event
termination when video starts
Att Overview : AUDIO - L/R
Spoken text with general information about Goya (female voice)

Figure 8 Representation model for the virtual museum application (extract).

7 CONCLUSION

With *HyDev* we have introduced a new model-based approach that supports the development of hypermedia applications. *HyDev* defines activities for each development phase. The development process starts with the modeling of the application's objects and their classes. In contrast to similar approaches *HyDev* works with additional special classes and relationships. The representation aspects (such as media object types, navigation, events, temporal and spatial aspects) are captured in a separate model. These specifications abstain from details and are refined during the specification&design phase.

Benefits of HyDev

Today one can often observe a „just make it" approach: HMAs are implemented rashly using authoring tools. *HyDev* however requires an initial determination and modeling of requirements. This way a developer is caused to consider and examine the application more thoroughly, resulting in a better understanding of the system to be built. Problems are identified early, and new ideas can evolve. As a consequence, the end product will be of higher quality concerning correctness, robustness and maintainability.

The various models are a mandatory basis for the activities of later development phases. In this respect, authors, designers, developers, and programmers are given a starting point for their work. Besides, the information captured in the models serve as an extensive documentation, especially for maintenance purposes (modifications, extensions, elimination of errors). Up to now, developers usually apply fairly informal documentation techniques like storyboards.

Certainly, the making of the various models costs time. But this extra effort pays off when it comes to implementation or maintenance of an existing product. Apart from that, a developer must think about these issues anyway. Currently, he/she does so on the side and disorderly, at worst during implementation. In any case it is better to analyze the intended application and capture the findings early and systematically.

Future work

So far, we have analyzed several existing products of various categories. *HyDev* was developed based on the resulting observations and findings. Currently, we are refining some aspects of the representation model. Furthermore, we are engaged in additional case studies, with the aim of validating our approach.

In the future, we will consider the activities of the specification&design and the implementation phase in more detail and develop appropriate models. In addition, we will provide a sophisticated edit tool offering extensive support for the creation and modification of the various *HyDev* models. Among the features of this tool will be elaborate drag&drop functionality, consistency checking, semantic zooming (the more we zoom in the more details can be seen), and selective viewing (e.g. only the directly connected neighbors of a selected object are shown; only objects of a specific class are shown). For that purpose we will adapt the generic edit tool *GenTool* which was developed in the context of our work on the FLUID-approach (Homrighausen et al., 1997; Kösters et al., 1996).

Finally, we will pay special attention to the issue of rapid prototyping. The rapid development of prototypes is especially advisable and realistic in the hypermedia domain. Since due to their complexity and abstractness the models are not suitable to be directly used for discussions with customers and/or users, we will examine how prototypes of HMAs can be generated (semi-)automatically on the basis of *HyDev* models.

8 REFERENCES

Booch, G., Jacobson, I. and Rumbaugh, J. (1997) *Unified Modeling Language (UML)*. Rational Software Corporation, Santa Clara, CA, version 1.1 edition

Garzotto, F., Paolini, P. and Schwabe, D. (1993) HDM - A model-based approach to hypermedia applications design. *ACM Trans. Info. Syst.*, 11, 1, 1–23

Gibbs, S. J. und Tsichritzis, D. C. (1995) *Multimedia Programming*, ACM Press Books, Addison-Wesley

Halasz, F., Schwartz, M. (1994) The Dexter Hypertext Reference model. *Communications ofthe ACM*, Vol. 37, No. 2, 30–39

Hardman, L, Bulterman, D. C. A., van Rossum, G. (1994) The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model. *Communications ofthe ACM*, Vol. 37, No. 2, 50–62

Homrighausen, A. and Voss, J. (1997) Tool support for the model-based development of interactive applications - The Fluid approach, *Proceedings of Eurographics Workshop on Design, Specification, Verification oflnteractive Systems*, DSV-IS '97, Eurographics Series

Isakowitz, T., Stohr, E. A. and Balasubramanian, P. (1995) RMM: A Methodology for Structured Hypermedia Design. *Communications ofthe ACM*, Vol. 38, No. 8, 34–43

Kösters, G., Six, H.-W. and Voss, J. (1996) Combined Analysis of User Interface and Domain Requirements, *Proceedings ofthe 2nd IEEE Int. Conf on Requirements Engineering*, Colorado Springs

Kozel, K. (1996) The Interactive Killing Fields. *Multimedia Producer*, No. 5, `http://www.kipinet.com/mmp/mmp_may96/feat_killing.html`

Madmind (1997) Madleine's mind. Interactive animated online comic, `http://www.madmind.com`

Schwabe, D. and Rossi, G. (1995) The Object-Oriented Hypermedia Design Model. *Communications ofthe ACM*, Vol. 38, No. 8, 45–46

9 BIOGRAPHY

Peter Pauen is a research assistant at FernUniversität Hagen. His research interests include hypermedia applications and software engineering.

Josef Voss is a research assistant at FernUniversität Hagen. He received a Ph.D. in computer science in 1990. His research interests include user interface tools and user interfaces in requirements engineering.

Hans-Werner Six is a full professor in computer science at FernUniversität Hagen since 1985. He received a Ph. D. from University Karlsruhe in 1978. His current research interests include Geographical Information Systems, Software Engineering, and graphical user interfaces.