OMMMA: An Object-Oriented Approach for Modeling Multimedia Information Systems

Stefan Sauer and Gregor Engels
University of Paderborn
Computer Science Dept.
D 33095 Paderborn, Germany
{sauer | engels}@uni-paderborn.de

Abstract

We present an object-oriented approach based on the Unified Modeling Language (UML) for modeling structure and dynamic behavior of a multimedia information system. We extend the Model-View-Controller paradigm for interactive systems towards multimedia and show that aspects of the graphical user interface and time-dynamic behavior ought to be integral parts of a model. This is done in the intention of achieving a coherent model of a multimedia information system. As a result, we present OMMMA-L, a UML-based language for the Object-Oriented Modeling of MultiMedia Applications. OMMMA-L deploys class and state diagrams of UML as well as an extended sequence diagram and a newly introduced presentation diagram to adequately specify the visual presentation within a multimedia information system. In addition to explaining the different diagram types, we also define pragmatic rules on how to deploy and combine the various diagrams. Additionally, we relate OMMMA-L to basic ideas of multimedia information systems and services.

1 Introduction

Multimedia software developers have to consider a wide range of aspects when building a multimedia information system. They do not only have to account for application structure and functionality, but also for user interface and system aspects like distribution, stream control, resource management or underlying database and network facilities. Due to the inherent complexity of such systems, the development should not be ad hoc, but has to be accompanied by a software development process like it is commonly accepted for traditional software and information systems and is demanded for multimedia applications as well. This development process must include an analysis and a design phase addressing specific multimedia application and system requirements.

This observation is even more striking if one analyzes common practice in multimedia software development. Existing development tools offer support for prototyping on the programming language level without producing appropriate documentation. Thus, multimedia applications are difficult to extend or adapt. Neither authoring systems like Macromedia Director or Asymetrix Toolbook nor specialized object-oriented multimedia frameworks for programming languages (see [10]), e.g. Java Media Framework [11], MET++ [1], explicitly support an analysis and design phase. Additionally, both approaches are by themselves not well-suited to describe architectural or other system-related aspects.

Research in the field of multimedia information and database systems has been dominated by system-related topics like content-based retrieval, multimedia query languages, storage management, buffer allocation, quality of service as well as playout scheduling and control in recent years (see [16, 5]). Aim of that work has been to develop database services that support multimedia applications based on distributed system architectures. In terms of a layered system architecture, this is a rather bottom-up approach.

In the context of the work presented here we address the problem from the opposite direction by analyzing what a multimedia information system application needs to know about the underlying system and the accessible services, especially in terms of the database management system. These aspects that are important for an implementation of a multimedia information system should then also be reflected in appropriate concepts and abstractions of the modeling language. This enables to account for specific system properties during analysis and design already.

Since we aim at an integrated model that accounts for interactivity and the temporal behavior of continuous media objects as well as a graphical multimedia user interface, the traditional techniques for database design and data modeling cannot simply be adopted for multimedia information systems. Several modeling concepts and languages have already been proposed to model specific aspects of multimedia applications and systems. The temporal characteristics of continuous media types are one of the most important features of multimedia systems.
Thus, different models have been dedicated to modeling temporal behavior and timing constraints (e.g. the timed Petri nets OCPN in [15] or the interval-based modeling presented in [23]). More elaborate models also account for interactivity (e.g. the timeline-tree model of [12]), since typical multimedia applications are expected to be highly interactive. Some of these models are rather technical, e.g. OCPN [15]. The MOAP approach presented in [24] is merely a textual specification language for class definitions and specialized temporal and spatial relations for multimedia objects and applications. The data model introduced in [7] is intended to structure time-series data and is coupled with an object-oriented database system and a visual querying language. Another focus of research has been the modeling of navigation in the context of hypermedia systems, e.g. OOHDIM [22] and RMM [13].

One of the most eminent shortcomings of all these approaches is that they are focused on special aspects of multimedia systems and do not offer facilities to model all important aspects of a multimedia system in an integrated, consistent, and coherent way. Therefore, they are not well-suited as a basis for straightforward implementation and do not enable consistency checks between different aspects on the modeling level. Software engineering perspectives to yield such an integrated model have not yet played such a prominent role like, e.g. object-oriented modeling approaches in the development of conventional software. Nevertheless, delivering appropriate concepts, languages, methods, and tools to model and implement the diverse aspects of this complex class of systems and consistently integrating the different system views are major challenges for research on multimedia information systems.

The objective of this work is therefore to develop a diagrammatic modeling language for multimedia information systems enabling an integrated specification of all relevant aspects of that system. We support the idea of developing a modeling language with diagrammatic notations on an intuitive level of abstraction, thus models can easily be understood by the different experts in multimedia development project who originate from diverse fields and show highly heterogeneous expertise in software development. We have selected an object-oriented approach based on the OMG standard Unified Modeling Language (UML; [17, 41]). Beyond the general advantages of the object paradigm for software modeling and implementation it has another advantage for multimedia information systems, since they are most appropriately based on object-oriented database systems. In addition to class and behavioral diagrams, UML also supports different implementation diagrams to model distribution and architectural concerns of the actual implementation. Storage and distribution aspects can thus be specified within an integrated model as well by further extensions.

Within this paper, the UML-based, visual multimedia modeling language OMMMA-L (Object-oriented Modeling of MultiMedia Applications - the Language) is introduced. Based on the characterization of central aspects of multimedia information systems, we have defined a software architecture model $\text{MVC}_{\text{MM}}$, extending the MVC model. The core of this paper is presented in Sect.3 where we point out essential elements of OMMMA-L. Furthermore, we state methodical guidelines on how to combine different model aspects to obtain an integrated model of the overall system. A brief discussion of OMMMA-L in the context of multimedia information system architecture is given in Sect.4.

2 Modeling the Characteristics of Multimedia Applications

We define a multimedia information system as an interactive, potentially distributed, multimedia application built on top of a multimedia database system that presents at least two combined media objects and shows time-dependent behavior. Media objects may be stored and accessed by a multimedia database management system.

A fundamental characteristic of multimedia applications is the composition of diverse aspects: In addition to the static features as regards content (building the focus of classical conceptual data modeling for database systems), the utilized media types, predefined and interactively controlled behavior as well as the design of the user interface must be taken into consideration. Besides that, aspects like the linkage to a database system or accessing distributed resources within a networked system are of importance for multimedia information systems. All these aspects are to be specified within an integrated and consistent model during the development of a multimedia application.

We can therefore identify the essential aspects of a multimedia application that are to be modeled as

- **logical structure** of application classes and their relations that represent application domain knowledge,
- **media types** that are associated with application classes and whose instances may persistently be stored in a database.
• predefined temporal behavior relating multimedia application objects in time, e.g. synchronization or sequentiality as intermedia or intra-media relations,

• spatial arrangement of the presentation manifested in different audio channels and the absolute and relative positioning of visual presentation objects on a presentation plane,

• interactive and event-driven dynamics, so-called spontaneous behavior in our approach, of which the temporal occurrence is not known in advance requiring reactive behavior, e.g. for selecting alternative flows of control or altering presentation.

This classification leads to a specific architecture of multimedia software that will be presented as an extension of the standard Model-View-Controller paradigm [14].

2.1 MVCMM: An Extended MVC-Model

Multimedia applications are interactive and have a strong emphasis on a multimedia user interface. Therefore, we considered software architectures for interactive systems as a basis for structuring and interrelating the aspects of a multimedia application. Two prominent models of this kind are Model-View-Controller and Presentation-Abstraction-Control. In contrast to a purely data- and functionality-oriented understanding of the modeling task as in traditional database (application) modeling, the advantage of these models is that they consider a much broader view of the system to be developed.

In the Presentation-Abstraction-Control (PAC; [6, 3]) approach, an application model consists of a hierarchy of cooperating agents, each with specific functional responsibilities. An agent consists of three facets, namely presentation, abstraction, and control. Since the PAC model does not explicitly distinguish between user input handling and presentation (both is associated with the presentation facet, whereas the control facet is responsible for the communication between its corresponding abstraction and presentation aspects as well as with other agents), we have chosen the Model-View-Controller paradigm as a basis for our modeling approach of multimedia information systems.

The Model-View-Controller (MVC; [14, 3]) paradigm is an architectural model widely known in object-oriented software development. It distinguishes a model component holding the core functionality (behavior) and data, a view component for displaying information to the user, and a controller component for handling user inputs. A change-propagation mechanism ensures consistency between the model on the one hand and the user interface components of the two types on the other hand (cf. [3]). If the controller receives a user input event to be handled, it informs the model of this occurrence by issuing a service request. The model executes the functionality associated with and triggered by this event and updates its representation of system state. Immediately afterwards, all associated controller and view components are informed about the updated system state, i.e. the change is propagated to them. MVC underlies many interactive systems and application frameworks for systems with graphical user interfaces, such as E1++ and ME1++ [1].

The separation of different concerns known from these architectural models was taken up for our multimedia modeling approach and supplemented by a further distinction of the model component into:

• the static application model consisting of the logical structure of application classes and a hierarchy of associated discrete and continuous media types, and

• the dynamic application model representing predefined temporal behavior.

This results in an extended Model-View-Controller model for multimedia, termed MVCMM (see Fig.1), with a refined and extended model component, a view component for the presentation aspect to support different presentations of a model component, e.g., realized by different presentation media, and, as in the MVC model, the control component, which is responsible for interactive and event-based control of both the application and the presentation. The separated controller component enables e.g. support for multimodal user interaction.

An important distinction between the explained MVC model and our MVCMM approach is the level of abstraction. (Therefore, we also use the term aspect instead of component.) While the MVC model is a pattern that is merely used during design and implementation phases, we distinguish the different aspects already on a more abstract level in the analysis phase. This results in a representation only of the distinctive properties of
3 OMMMA-L: The Language

In this section, we will introduce our modeling language OMMMA-L by showing how its diagram types extend the standard object-oriented modeling language UML appropriately. OMMMA-L allows to model all aspects of a multimedia application according to the extended \( MVC_{MM} \) model in an integral and coherent form.

The Unified Modeling Language (UML; [17, 4, 18]) consists of a set of diagram languages which are tailored to specify distinguished aspects of a system to be modeled. As it turns out from our investigation, the logical structure of an application as well as interactive control and parts of temporal behavior can be adequately modeled with UML. But specialized and more advanced language constructs are needed to describe the precise temporal ensembling of different media objects. Additionally, an explicit notation is needed for spatial modeling in order to intuitively specify user interface layouts. Finally, UML lacks appropriate pragmatic guidelines on how to deploy the different diagram types cooperatively to model complex multimedia applications. These shortcomings lead to the development of an extension of UML.

Within the following subsections, we give an overview of the diagram types of OMMMA-L followed by a description of their combination. A more detailed description of OMMMA-L, accompanied by a sample application illustrating the use of the different diagram types and language features to model a multimedia encyclopedia, can be found in [19].

3.1 Class Diagram

Class diagrams are the core of an object-oriented application model and are used to describe the static part of the model aspect of the \( MVC_{MM} \) model. Essentially, they consist of class and association definitions which describe the structure of objects and their possible structural interrelations. Language features of UML class diagrams have been incorporated unchanged into OMMMA-L. An OMMMA-L class diagram consists of two closely interrelated parts in order to express the two static model aspects \textit{application logic} and \textit{media types} of an application:

- An \textit{hierarchy of media type definitions}, which comprises classes for all discrete and continuous media types.

- the \textit{logical model} of an application, which comprises classes and associations to describe application domain objects and their relationships.
The two aspects are linked by associations interrelating application and media objects. We explicitly distinguish between application objects as regards content and media objects to allow for presenting one application entity by different media.

3.2 Extended Sequence Diagram

UML offers various diagram types to model behavioral aspects of an application. Due to their emphasis on modeling time constraints, sequence diagrams are deployed in OMMMA-L to model the predefined temporal behavior of a multimedia application, i.e., the dynamic part of the model aspect of the $MVC_{MM}$ model. But in order to supply appropriate language abstractions to model specific characteristics of a multimedia application that are frequently needed, standard UML sequence diagrams have been extended by a series of features (see [19]), e.g., refinement of the time axis by time stamps and intervals, sequence diagram parameterization, activation and deactivation delays, simultaneous activation clustering as well as sequential activation composition, and time-dependent media filters superimposed on activations.

An example of an OMMMA-L sequence diagram is given in the lower right of Fig.2. It describes the execution of Music(int M idle) which presents a certain piece of a music accompanied by a dynamically highlighted sheet of music. Delay constraints and media filters are described by appropriately shaded areas within the activation boxes.

A sequence diagram models a predefined presentation sequence within a scene. All objects in one diagram relate to the same timeline (intermedia synchronization). The relation of a single continuous media object to the corresponding timeline specifies inmedia synchronization. Concurrent activations with an independent timeline need to be modeled by different sequence diagrams related to parallel subtrees within an and-superstate of the corresponding state diagram (see Sect.3.4). Timing marks and duration intervals can be given absolute or relative to other activations. A propagation mechanism has to ensure consistency of temporal specifications.

3.3 Presentation Diagram

Due to the fact that UML does not offer a diagram type which is well-suited for intuitively modeling the spatial composition structure of the presentation (view aspect of $MVC_{MM}$), the new presentation diagram type has been added to OMMMA-L. The presentation diagram allows a static specification of presentation layout. In addition, by incorporating the user interface design into the modeling language, consistency relations to other diagram types can be formulated and checked.

The presentation diagrams of OMMMA-L are composed of sized, possibly overlapping bounding boxes that are positioned on a virtual area. Presentation objects that are assigned to these bounding boxes are distinguished into visualization objects and interaction objects (depicted by bounding boxes with heavy borders) depending on whether they allow for user interactions and may raise events in the running system.

The complete user interface layout of a certain application unit may be described by several presentation diagrams, called layout views being composed by a layered placement on the virtual area. The lower left part of Fig.2 gives an example of an OMMMA-L presentation diagram composed from three layout views. OMMMA-L offers a variety of predefined specializations of bounding boxes to indicate the type of object to be presented or to indicate frequently-used user interaction elements. The visual layout specification is accompanied by an iconic representation of audio channels beside the visual plane.

3.4 State Diagram

While sequence diagrams are used to specify the predefined behavior as the dynamic model aspect, state diagrams are deployed to specify behavior triggered by user interactions or other system events, i.e., the spontaneous behavior represented in the controller aspect of $MVC_{MM}$. With a single exception, OMMMA-L state diagrams are syntactically identical to UML state diagrams, i.e., they may be structured by composite states or refined by embedded state diagrams. One specific syntactic alteration has been added: In order to couple the interactive control with the predefined parts of a multimedia application, internal actions of simple states may be labelled by names of sequence diagrams instead of embedded state diagrams or action expressions. This means that the behavior specified by the sequence diagram is automatically triggered whenever the corresponding state has been entered. This construction is feasible since action expressions in UML can be specified by sequence diagrams as well as OMMMA-L sequence diagrams can be transferred into state diagrams on the multimedia application level.
3.5 Combining the Diagram Types

Each of the above introduced OMMMA-L diagram types is used to specify a certain aspect of a multimedia application. According to the refined MVCM model, diagram types are associated to aspects as follows: Class diagrams are used to specify the static model aspect of \(MVCM\), sequence diagrams for the dynamic model aspect, presentation diagrams for the view aspect, and state diagrams to specify the controller aspect.

In a more detailed view, a specification of a multimedia application consists of a series of multimedia application units, so-called *scenes*. Each scene corresponds to a state within an overall state diagram which is associated to the complete application specification. Furthermore, each scene is related to a presentation described by a complete presentation diagram possibly composed from multiple layout views. A state associated to a scene may be refined by a further state diagram, which describes the interactive behavior within this scene. Atomic states are associated to sequence diagrams which represent the predetermined scenarios of behavior within a scene. Application objects that are to be presented at the user interface are assigned to a corresponding bounding box or audio channel via an activation within a sequence diagram. The activation can reference a media object (the interrelations to media types are simplified to superclasses in Fig. 2) that is being presented during this activation.

Figure 2 gives an example of such a complete specification, where small parts of each diagram type are depicted. Diagrams are interrelated by using the same identifier names in different diagrams. Examples are the name of a sequence diagram as action expression of an internal action within a simple state of the state diagram, or the name of a bounding box or audio channel of a presentation diagram within a sequence diagram associated to an activation box. Other consistency constraints between diagrams are depicted by overlaid arrows.
The context-free syntax of diagram types as well as those context-sensitive constraints for the interrelation between diagram types of OMMMA-L have been defined by refining the UML meta model. General constraints from the UML meta model have been specialized to express the OMMMA-L specific usage and interrelation of diagram types. Furthermore, stereotypes are used to describe new and refined language features according to the standard extension mechanisms of UML. The OMMMA-L meta model can be found in [20].

4 OMMMA-L and Multimedia Information Systems

After presenting an overview of the OMMMA-L language elements and their deployment and combination on an abstract level, we will now discuss the relations to multimedia information systems. As stated earlier, the static model as specified in the class diagram relates to object-oriented data modeling. The media type hierarchy builds the connection to data storage. Instances of these classes reference to and load database entities holding the actual media. In a client/server-architecture, they relate to the database server, while the application classes relate to the database client. Application classes define the external schema (view) of the multimedia application.

The predefined temporal behavior represented in OMMMA-L sequence diagrams resembles the specification of a presentation plan specifying the playout scheduling for the multimedia application. For instance, in the AMOS client/server-architecture, a prototypical multimedia database system (see [2]), it is computed by the multimedia database server for presenting the results of a query and executed by a database client running the application. If we regard a query as one possible user interaction, it can be mapped to a transition in the OMMMA-L state diagram, where the entered state specifies the presentation plan to be executed. Since database applications abstract from concrete instances, it is important to stress that OMMMA-L sequence diagrams can also be used on this specification level (in contrast to the instance level) and that the parameterization of sequence diagrams adds to a flexible specification.

The control and view aspects of MVC\textsubscript{MM} can be assigned to the multimedia application running on a database client. In traditional database application development, they are merely considered on the level of implementation.

5 Conclusions

We have presented OMMMA-L, a visual, object-oriented modeling language for multimedia information systems extending the standard modeling language UML. New language features have been incorporated into OMMMA-L in order to allow modeling of all aspects of a multimedia application. OMMMA-L is based on an extended Model-View-Controller (MVC) model for multimedia applications, termed MVC\textsubscript{MM}.

The conceptual work on designing a new modeling language is accompanied by the realization of a first prototype implementation OMMMA-Tool, extending the commercial CASE tool Rational Rose98 by additional visual editors for the new, resp. extended diagram types of OMMMA-L according to the defined pragmatic guidelines. The editors are syntax-directed and ensure the consistency of the different diagrams within a model. Rational Rose98 is also deployed as the repository for managing model data.

Currently, OMMMA-L is evaluated in different scenarios, as e.g. in an industrial project for the specification of multimedia components within automobile cockpits. Furthermore, it is investigated whether more evolved features (cf. [8, 9]) for the specification of media object synchronization have to be incorporated into OMMMA-L. In a parallel activity we are developing extensions to OMMMA-L to account for system characteristics and requirements.

References


