Extending UML for Modeling of Multimedia Applications

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

Abstract

An analysis of how visual modeling of structure and dynamic behavior of a multimedia application differs from modeling conventional software yields that aspects of the graphical user interface and time-dynamic behavior ought to be integral parts of a coherent multimedia application model. In this sense, we extend the model-view-controller paradigm towards multimedia. As a result, we present OMMMA-L, a visual Language for the Object-Oriented Modeling of MultiMedia Applications that is based on the Unified Modeling Language (UML). The structural and behavioral diagram types of UML have been analyzed and are adapted and extended according to multimedia application characteristics. A presentation diagram is introduced and integrated to adequately describe the visual presentation. In addition to explaining the different diagram types, we also give pragmatic guidelines on how to deploy and combine the various diagrams.

1. Introduction

Multimedia applications are interactive software systems in which objects of diverse discrete and continuous media types are combined and presented together. We define a multimedia application as an application that combines at least two media objects and shows time-dynamic behavior. Such an application might use objects of continuous media types, i.e., media possessing a time-variant presentation like animation, video or audio, or a sequence of pictures in a slide show in which each picture is presented for a specific period of time.

Interactive multimedia applications will become a widely-used kind of future software systems. By integrating multimedia elements, application programs can be made more comprehensible, especially if the information to present is itself inherently multimedia. Additionally, multimedia can also make user interaction more intuitive by reproducing natural forms of interaction, e.g. with simulated laboratory instruments like rotary switches or analogue indicators, or by speech and gestures [2].

State of the art in building multimedia applications is to deploy specialized multimedia frameworks or authoring systems. Frameworks and toolkits, e.g. Java Media Framework [6] and MET++ [1], are integrated into programming languages and environments to allow for linking multimedia elements in application programs. Authoring systems like Macromedia Director or Asymetrix Toolbook are orientated towards a rapid ad hoc application development offering direct-manipulative interaction, supplemented by a scripting language containing simple language constructs to program extended functionality by hand. In both cases, no explicit support for an analysis and design phase is offered.

From this observation, we claim that the development process of multimedia applications should include an analysis and a design phase like they are essential in conventional software engineering. Especially the design phase is required to achieve a clearly structured and error-free implementation. Concepts, languages, methods, and tools must be developed that take the specific requirements of multimedia systems into account and support a methodical multimedia software process. Within this work, we focus on a modeling language for multimedia that enables an integrated specification of all relevant aspects.

A number of modeling approaches have been proposed for multimedia applications. Predominantly, they focus on modeling of temporal relations and synchronization of multimedia presentations (see e.g. [10, 16]). Some more elaborated models also account for interactivity (e.g. [7]). Others concentrate on logical structure and navigational concepts for hypermedia (e.g. [8, 15]).

Software modeling languages are normally based on the object paradigm today. Object-orientation offers a uniform concept for software development and numerous advantages like the integrated specification of structure and behavior as well as the seamless integration through all development phases. In recent years, several object-oriented
modeling languages have emerged of which the Unified Modeling Language (UML; [12, 3]) is one of the latest and most important. UML has become an OMG standard and a quasi-standard in industrial software development. It is a visual, diagrammatic specification language and facilitates the specification of both software systems and business processes allowing a uniform treatment of applications and the process models for their development.

Unfortunately, UML does not support all aspects of multimedia applications in an adequate and intuitive manner (see Sect. 3.1). Especially, language features for modeling user interface aspects are not explicitly provided. Other concepts of UML are not mature enough or less vivid and thus aggravate multimedia modeling unnecessarily.

The aim of this work is therefore to introduce the UML-based, visual multimedia modeling language OMMMA-L (Object-oriented Modeling of MultiMedia Applications - the Language). In this intention, we first analyze the characteristics of multimedia applications and evolve our approach of their modeling in Sect. 2. The core of this paper is presented in Sect. 3 where we point out the essential elements of OMMMA-L for specification of the different aspects of multimedia applications. Section 4 concludes this article by summarizing the achievements and ongoing work as well as drawing some future perspectives.

2. Modeling the Characteristics of Multimedia Applications

A fundamental characteristic of multimedia applications is the composition of diverse aspects: In addition to the static features as regards content, the utilized media types, the predefined temporal behavior and the interactively controlled behavior as well as the design of the user interface including the spatial layout of the presentation must be taken into consideration. Besides, aspects like the linkage to a database system or accessing distributed resources within a networked system, e.g. the connection to the Internet or mobile communications, are of importance, too. All these aspects are to be specified within an integral and consistent model during the development of a multimedia application. Within this work, we start with the central aspects and focus on the logical structure, the spatial presentation layout, the predefined temporal behavior, and the interactive control.

2.1. Sample Application: Classical Composers Encyclopedia

As an example of an interactive multimedia application we assume an encyclopedia containing information on classical composers (see Fig. 1).

A composer entry is a composition of several items. A textual description is provided as a hypertext. Hyperlinks allow to invoke the presentation of specific media objects. Images and sheets of music can be viewed, music pieces or video tracks can be played. These application objects are associated with media objects. The user interface layout for visual presentation and the possible user interactions are realized uniformly for every entry throughout the application. Several information and control items are placed on the screen. Interactive application control can be done in different ways: The last name initial can be selected by mouse-clicking on an alphabetic menu bar. An animated arrow points to the presently selected character. When selected, the composer list presented within a hypertext window with scrollbar is skipped to the first entry beginning with the corresponding letter. There, the composer can be directly selected. A media control component allows the control of continuous media. An end button and buttons for skipping forward and backward between entries complete the application.

2.2. Aspects of Multimedia Applications

In this subsection, we will discuss the four central aspects of a multimedia application in more detail.

(1) Logical Structure

The logical structure of a multimedia application is an independent aspect and must be specified explicitly. As in conventional software engineering processes, it is the starting point for application modeling. The logical dimension represents the application domain knowledge. For the multimedia domain, this model must not only contain classes defining the structure of application objects and the relationships among them, but also their association to instances of specific media types. Particularly, it has to be distinguished whether they are associated to discrete or continuous media type instances.
Spatio-temporal Presentation

Multimedia applications have a spatio-temporal presentation. This presentation can be further distinguished into a visual presentation and an audio presentation. We partition the modeling of the presentation component into a temporal and a spatial dimension that are related to each other. The temporal dimension is independent of the presentation mode (audio or visual).

(2) Spatial Presentation

The visual and auditory presentation of media is an integral part of the application. The spatial dimension of the audio presentation is manifested in different audio channels. Layouting the visual user interface includes the absolute and relative positioning of visual presentation objects as well as their composition. The positioning is concerned with object position in a virtual space and their projection onto a presentation plane like a computer screen. An explicit modeling of the spatial layout is necessary to be capable to specify uniform presentation of different objects as part of an integral application model.

(3) Predefined Temporal Behavior

The behavior of a multimedia application is determined by both predefined temporal sequences and user interaction, e.g. user-initiated navigation or manipulation of presentation streams of continuous media objects. Because of that, it is useful to distinguish between predefined time-dynamic behavior hard-wired within the application and spontaneous, i.e. interactive and event-driven dynamics of which the temporal occurrence is not known in advance.

To model predefined temporal behavior, it must be possible to relate objects of the multimedia application in time. Typical temporal relations are synchronization or sequentiality. These can be specified either relative to other objects (inter-media relation) or absolute regarding a metric time axis (intra-media relation). Especially for continuous media types, an ongoing synchronization with a real clock must be ensured to a high degree, e.g. to prevent jitter in videos. Predefined temporal relations should be intuitively represented based on a visual time representation.

(4) Interactive Control

User interactions or other events raised by the application environment require reactive behavior of the application. Based on such events, alternative flows of control may be chosen or the presentation may be altered.

2.3. $MVC_{MM}: An Extended MVC-Model$

Multimedia applications are interactive and have a strong emphasis on a multimedia user interface. Therefore, we considered software architectures from the field of user interface modeling as a basis for structuring and interrelating the aspects of a multimedia application explained above. A similar approach was also followed in the development of the MET++ framework [1]. This is in contrast to usually followed development approaches, where the development of the user interface is postponed to the design or even implementation phase where GUI builders or authoring systems are deployed for an immediate, ad hoc realization of user interfaces.

The separation of different concerns known from these software architecture models for interactive systems, e.g. Model-View-Controller (MVC; [9]) or Presentation-Abstraction-Control (PAC; [4]) was taken up and supplemented by a further distinction of the model (abstraction, resp.) component into

- the static application model, consisting of the application logic and a hierarchy of discrete and continuous media types, and
- the dynamic application model.

This resulted in an extended Model-View-Controller-Model for multimedia, termed $MVC_{MM}$ (see Fig. 2), with a refined and extended model component, a view component for the presentation aspect, and, as in the PAC and MVC models, the control component which is responsible for controlling both the application and the presentation. Its separation also allows to account for multimodal interaction.

3. OMMMA-L: The Language

In this section, we will introduce our modeling language OMMMA-L. We will show how this language extends the standard object-oriented modeling language UML appropriately and allows to model all aspects of a multimedia application according to the extended $MVC_{MM}$ model in an integral and coherent form.

3.1. The Unified Modeling Language UML

The Unified Modeling Language (UML; [12, 3, 13]) consists of a set of diagram languages which are tailored to specify distinguished aspects of a system to be modeled.
Those are grouped into four categories: use case diagrams, structural diagrams, behavioral diagrams, and implementation diagrams. In the following, we focus on static structure and dynamic behavior of a multimedia application and therefore analyze only the second and third of these diagram categories. UML allows to specify structural aspects in class and object diagrams. Behavior can be described using sequence, collaboration, state, and activity diagrams. As it turns out, the logical structure of an application as well as interactive control, accompanied by a tailored dialog signal hierarchy, and parts of temporal behavior can be adequately modeled with UML. But specialized and more advanced language constructs are needed to describe the temporal ensembling of different media objects. Additionally, UML needs to be extended by an explicit notation for spatial modeling in order to specify user interface layouts intuitively. 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 towards capturing these application characteristics and to derive pragmatics on how to model multimedia applications with an object-oriented language based on UML.

3.2. 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 MVCMM model. Essentially, they consist of class and association definitions which describe the structure of objects and their possible structural interrelations. As UML’s language features for defining a class diagram are expressive enough, they have been incorporated unchanged into OMMMA-L. But in order to express the two static model aspects application logic and media types of an application, each OMMMA-L class diagram consists of two closely interrelated parts:

- An hierarchy of media type definitions, which comprises classes for all media types, and
- the logical model of an application, which comprises classes and associations to describe application domain objects and their interrelations.

The two aspects are linked by associations which interrelate application objects with corresponding media objects. Figure 3 gives the class diagram for the running example. The left-hand part of the diagram shows the media type hierarchy and the right-hand part the model of the logical structure of the application domain. It shows that each composite entry is a complex composition of elementary entries, where each of them is associated with one or more specific media types (e.g. audio or video). We explicitly distinguish between application objects as regards content and media objects to allow to present one application entity by different media. Thus, media objects are not specializations of application objects or vice versa. The dimension of media types is based on a generalization hierarchy of discrete and continuous media types as it can be found in several multimedia standards, e.g. MHEG [11], and frameworks, e.g. the framework of Gibbs and Tsichritzis [5] or MET++ [1].

3.3. Presentation Diagram

Class diagrams are used to describe the static part of the model aspect of our MVCMM approach. Before we continue with dynamic aspects in the next subsections, we first explain how another static aspect, the structure of the presentation (view), is modeled in OMMMA-L.

Due to the fact that UML does not offer a diagram type which is well-suited and appropriate for this task, the new presentation diagram type has been added to OMMMA-L. Presentation diagrams allow an intuitive description of the layout, i.e. the spatial arrangement of visual objects at the user interface. 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 follow the idea of structuring the user interface by bounding boxes. Bounding boxes have a certain size and are positioned on a virtual area. Objects that are to be presented at the user interface are assigned to a corresponding bounding box. Those objects are distinguished into visualization objects and interaction objects. Visualization objects are passive objects at the user interface and are used to present e.g. text, pictures, graphics, video, or animation objects at the user interface. Interaction objects allow for user interactions and may raise events in the running system. Examples are scroll or menu bars, buttons, input fields or a hypertext containing links. The visual layout specification is accompanied by an iconic representation of audio channels beside the visual plane.

Figure 4 gives an example of an OMMMA-L presentation diagram. A left and a right audio channel are depicted by icons on the left. Several bounding boxes are positioned on the virtual area. Each bounding box has a certain name, which has to be unique within a presentation diagram. Bounding boxes for interaction objects are indicated by bold borders, as it is done in UML with active objects. OMMMA-L offers a variety of predefined specializations of bounding boxes like animation, video or button, scroll bar, etc. to indicate the type of object to be presented in a bounding box. For instance, buttons which control the flow of the application or the presentation of assigned continuous media objects are indicated by a black bullet.

The complete user interface layout of a certain appli-
A sequence diagram models a predefined presentation sequence within a scene. All objects in one diagram relate to the same timeline (inter-media synchronization). The relation of a single continuous media object to the corresponding timeline specifies intra-media synchronization. Concurrent activations with an independent timeline need to be modeled by different sequence diagrams related to parallel sub-states within an and-superstate of the corresponding state diagram (see Sect. 3.5). Different message types between objects enable specification of synchronous or asynchronous messaging. Timing marks and duration intervals can be given absolute or relative to other activations. A propagation mechanism has to ensure consistency of temporal specifications.

Figure 5 gives an example of an OMMMA-L sequence diagram. It describes the execution of `Music(int MTitle)` which presents a certain piece of a music sheet in the corresponding bounding box on the screen and plays a certain piece of music on the left and right audio channels, while marking the currently played music parts by highlighted boxes on the screen. The interrelation between the activation of objects and the presentation diagram is indicated by using the same identifiers. Associated media objects that must conform to the types specified in the class diagram are enclosed in `<>`. Delay constraints and media filters are described by appropriately shaded areas within the activation boxes.

3.4. 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. But, in order to be able to model specific characteristics of a multimedia application more directly and thus more intuitively, standard UML sequence diagrams have been extended by a series of features. These are for example:

- **Refinement of the time axis** by introducing time stamps as well as different forms of determined, bounded or undetermined time intervals.
- **Parameterization** of sequence diagrams, e.g. by time stamps for start and end of execution of a sequence diagram, in order to support their reuse, or to be used in guard expressions of messages.
- **Activation and deactivation delays** of media objects in order to model tolerated variations of synchronization constraints for media objects (compare maximum start and end times in [7]).
- **Composed activation** of media objects in order to model the clustering of concurrently active objects.
- **Different activations** may be sequentially composed resulting in an **automatic triggering** of subsequent activations.
- **Activations of objects** may be overlayed by **media filters**, which describe time functions, e.g., the incremental increase of an audio level over time.

Figure 3. OMMMA-L class diagram
3.5. State Diagram

While sequence diagrams are used to specify the predefined behavior of a multimedia application, state diagrams are used to specify the system states as well as state transitions triggered by user interactions or other system events, i.e. the spontaneous behavior. Basically, OMMMA-L state diagrams are syntactically equal to UML state diagrams. This means that, e.g., 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, non-interruptable parts of a multimedia application, internal actions of simple states may be labelled by names of sequence diagrams instead of embedded state diagrams. This means that the behavior specified by the sequence diagram is automatically triggered whenever the corresponding state has been entered. Figure 6 gives an example. It shows a part of a state diagram where within the state PlayMusic as internal action the sequence diagram labelled Music is activated. This construction is feasible since OMMMA-L sequence diagrams can be transferred into state diagrams on the multimedia application level.

Mutually exclusive substates of an or-superstate enable the specification of alternative presentation flows triggered by events on incoming transitions comparable to the timeline-tree model in [7].

To allow for dialog specification on an adequate level of abstraction, e.g. the selection of navigational alternatives or the control of media playout, the OMMMA-L application model needs to be accompanied by an appropriate signal hierarchy (see [12]). This must be based on a spatio-temporal event model for user interactions on the control interface. It should, in perspective, also account for modal parameters of event instances.

3.6. 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 $MV_{CM}$ model, diagram types are associated to aspects as follows:

- Class diagrams are used to specify the logical structure of an application including the association of application objects to media types, i.e. the static part of the model aspect of $MV_{CM}$.
- Presentation diagrams are used to specify the spatial presentation, i.e. the view aspect of $MV_{CM}$.
- Sequence diagrams are used to specify the predefined temporal behavior, i.e. the dynamic part of the model aspect of $MV_{CM}$.
- State diagrams are used to specify the interactive control, i.e. the controller aspect of $MV_{CM}$.

Specification of a multimedia application may be more detailed a collection 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 of different layout views. A state associated to a scene may be refined by a further state diagram, which describes the interactive behavior within this
Atomic states are associated to sequence diagrams which describe the predefined, non-interruptable pieces of behavior within a scene. An activation of an application object can reference a media object (the interrelations to media types are simplified to superclasses in Fig. 6) that is being presented during this activation.

Figure 6 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 overlayed arrows.

The context-free syntax of diagram types as well as the context-sensitive constraints for the interrelation between diagram types have been defined for OMMMA-L by the use of a so-called meta model. In fact, the OMMMA-L meta model is a refined UML meta model where general constraints from the UML meta model have been specialized to express the OMMMA-L specific usage and interrelation of diagram types. Furthermore, stereotypes have been used to describe new language features and diagram extensions. Due to space limitations, the OMMMA-L meta model is not contained within this paper. It can be found in [14].

4 Conclusions

We have presented OMMMA-L, a visual, object-oriented modeling language for multimedia applications. OMMMA-L is based on the standard modeling language UML. New language features have been incorporated into OMMMA-L in order to allow integrated modeling of all aspects of a multimedia application. Particularly, a presentation diagram type and appropriate extensions to sequence diagrams have been introduced. OMMMA-L is based on an extended Model-View-Controller (MVC) model for multimedia applications, termed $MVC_{MM}$. This takes into account that aspects like the application logic, the used media types, the user interface as well as their controlled behavior have to be modeled coherently.

The conceptual work on designing a new modeling language, as it is presented in this paper, is accompanied by the realization of corresponding support tools. A first prototype implementation OMMMA-Tool has been accomplished extending the commercial modeling tool Rational Rose 98 by additional visual, syntax-directed editors for the new, resp. extended diagram types of OMMMA-L.

Currently, OMMMA-L is evaluated in different scenarios, as e.g. in an industrial project for the specification of multimedia information services within automotive cockpits. Furthermore, is is investigated whether more elaborate features for the specification of synchronizing objects have to be incorporated into OMMMA-L to specify the synchro-
Figure 6. An integrated OMMMA-L specification

Organization of media objects in a more detailed way. Additionally, we intend to define a formal model for the composition within and between the different behavioral diagrams.

References


