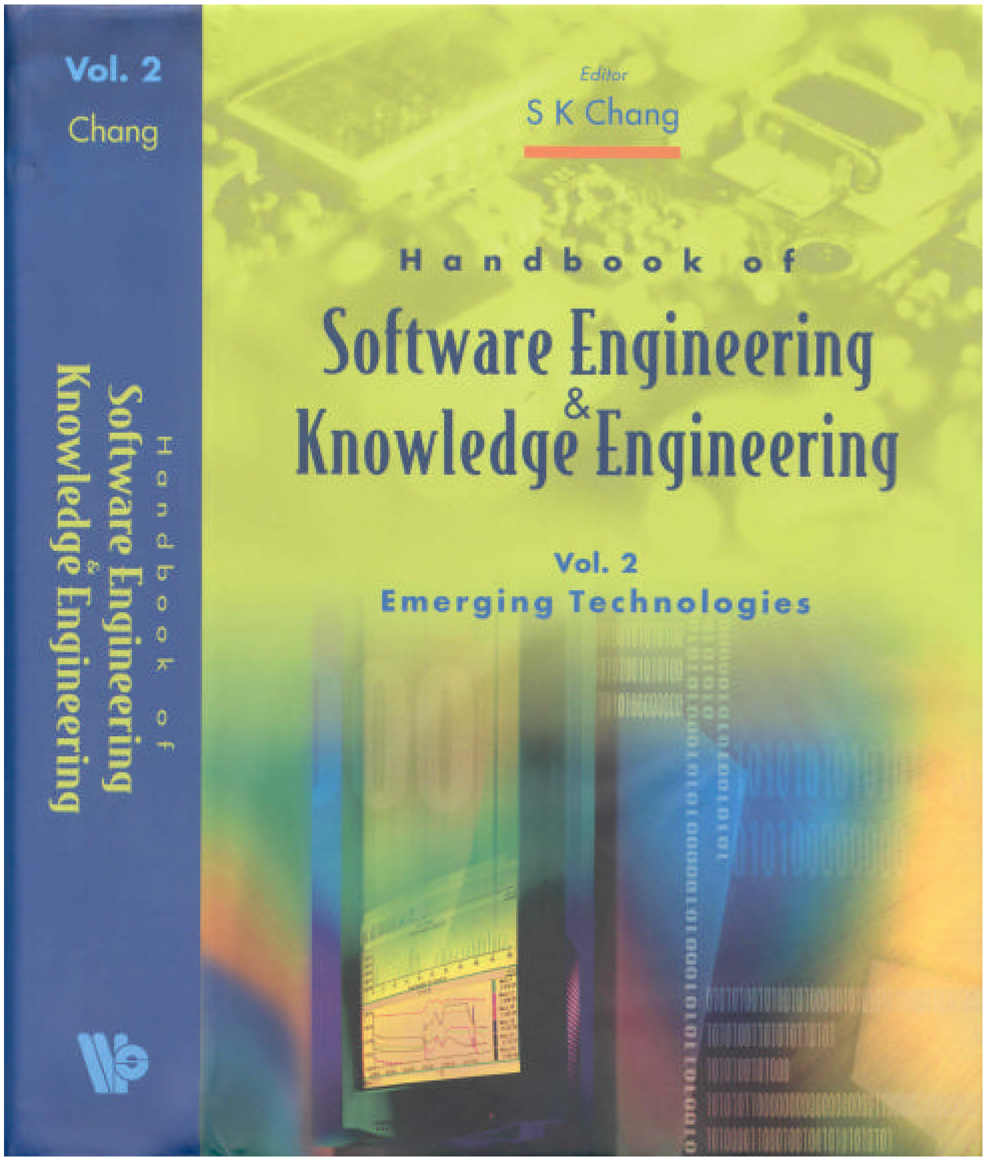
HANDBOOK OF SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING

edited by S K Chang *(University of Pittsburgh, USA & Knowledge Systems Institute, USA)*


Vol. 2
Chang
Handbook of Software Engineering & Knowledge Engineering
Editor
S K Chang
Handbook of
Software Engineering & Knowledge Engineering
Vol. 2
Emerging Technologies

Gregor Engels and Stefan Sauer

Object-oriented Modeling of Multimedia Applications.
In S.K. Chang (editor),
Handbook of Software Engineering and Knowledge Engineering,
vol. 2, pp. 21-53, World Scientific, Singapore, 2002.

Vol 1: Fundamentals	ISBN 981-02-4973-X	940p
Vol 2: Emerging Technologies	ISBN 981-02-4974-8	808p
Set	ISBN 981-02-4514-9	

http://www.wspc.com/books/compsci/4603.html

OBJECT-ORIENTED MODELING OF MULTIMEDIA APPLICATIONS [*]

GREGOR ENGELS and STEFAN SAUER
University of Paderborn
Mathematics and Computer Science Department
D–33095 Paderborn, Germany
E-Mail: {engels|sauer}@upb.de

The field of multimedia software engineering is still in an inmature state. Significant research and development has been dedicated towards multimedia services and systems technology such as networking or database systems. Multimedia document formats have been standardized. But when it comes to multimedia application development, the development process is truncated to an implement-and-test method. Either specialized multimedia authoring systems or multimedia frameworks or toolkits complementing programming languages or system software are directly used for implementation. No preceding modeling phases for requirements specification, analysis, or design of the system to build are enforced. The development of sophisticated multimedia process models and established, usable graphical notations tailored to the specification of multimedia systems is still underway.

In order to fill this gap, it is the purpose of this chapter to show current achievements in object-oriented modeling of multimedia applications. Based on an analysis of the state of the art in multimedia application development, we shortly present approaches to object-oriented hypermedia modeling and extensions of the Unified Modeling Language (UML) for hypermedia and interactive systems. The main part of the chapter is dedicated towards a recent approach to the Object-oriented Modeling of MultiMedia Applications (OMMMA).

Keywords: Multimedia software engineering, object-oriented modeling, integrated modeling, Unified Modeling Language.

[*] chapter in S. K. Chang (editor), *Handbook of Software Engineering and Knowledge Engineering*, vol. 2, pp. 21-53, World Scientific, Singapore, 2002. ISBN 981-02-4974-8. http://www.wspc.com/books/compsci/4603.html

1 Introduction

Multimedia applications can be defined as interactive software systems combining and presenting a set of independent media objects of diverse types that have spatio-temporal relationships and synchronization dependencies, may be organized in structured composition hierarchies, and can be coupled with an event-based action model for interaction and navigation. Media objects contribute essentially to presentation and interaction via the user interface. Two major categories of media types can be distinguished:

- *static media types*: time-independent media that do not show any temporal expansion and whose presentation is invariant over time, e.g. text, image, and graphics; and
- *temporal media types*: time-dependent media that possess time-dynamic behavior and whose presentation varies over time, e.g. animation, video, and audio.

Interactive multimedia applications are becoming a widely-used kind of software systems. By integrating multimedia elements, application programs can be made more comprehensible. For instance, it is better to provide auditive examples of musical pieces within an encyclopedia of classical composers or to show a video on the deployment and functioning of a computer tomograph than to simply provide this information textually or with single pictures. Therefore, multimedia is especially useful 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 [9]. We expect that typical multimedia application domains will not be restricted to purely presentational objectives, like web pages, educational courseware, interactive entertainment, or multimedia catalogues, but conventional business, technical, and information systems will be extended with multimedia features.

State of the art in multimedia software development is that multimedia applications are directly built either by deploying multimedia authoring systems or by coding them using multimedia frameworks or toolkits adding multimedia features to (object-oriented) programming languages or system software. In both cases, no preceding modeling phase as part of a sophisticated software engineering process is carried out prior to implementation. But the importance of such an activity in the course of multimedia software development has been advocated (see e.g. [3] or [8]).

From a software engineering perspective, the problem with the current state of multimedia application development is not only the absence of *sophisticated, yet practical multimedia software development process models*, but also the lack of *usable (visual) notations to enable an integrated specification* of the system on different *levels of abstraction* and from different *perspectives*. Such languages must be understandable by the different people and stakeholders involved in the multimedia development process.

While support by authoring systems, multimedia technologies, services, and standard formats eases the implementation and exchange of multimedia software, they must be complemented by more abstract, yet precise modeling or specification methods for these systems. What is especially missing is the integrated specification of the multiple aspects of multimedia systems like real-time behavior, user interaction, and application logic. This becomes a key issue when multimedia applications continue to evolve towards higher complexity, and multimedia features become prominent in almost any application domain in order to make information more conceivable as well as user interaction more sophisticated and lively. Effective management and maintenance of such applications requires testing and quality assurance, adaption and reuse of its parts, as well as readable and structured documentation. These requirements are supported by a modeling activity within the development process. Precise specification techniques add to this the capability of validation other than testing by model analysis. Instead of authoring systems supporting the instance-level assembling of interactive multimedia presentations, multimedia CASE tools are needed that support an integrated development process. The implement-and-test paradigm used during multimedia authoring resembles the state of software development before leading to the software crisis of the 1980s.

From this observation, we claim that the development process of multimedia applications should include a *modeling* activity, predominantly on analysis and design levels, like they are essential in conventional software engineering methods. 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. As regards software specification prior to implementation, we consider an *integrated visual modeling language* with capabilities to describe all relevant aspects of the system in an interrelated fashion as the most promising approach. Since the object paradigm has proven many intrinsic properties to support software product and development process quality – like encapsulation, modularity, extensibility, adaptability, portability, integrated specification of structure and behavior, or seamless integration by using a uniform paradigm throughout the process – we regard an *object-oriented modeling language for multimedia* that enables an integrated specification as being well-suited for this task. Object-oriented modeling eases the transition to the implementation of a multimedia application since implementation technologies as well as multimedia databases are mostly based on the object paradigm, too.

A number of models have been proposed for multimedia applications. One striking disadvantage of most models is that they only support partial models for individual aspects of the system, but not a holistic model for the integration of these modeling dimensions. Primarily, they focus on modeling of temporal relations and synchronization of multimedia presentations (e.g. [41, 61]; consult [6] for a general classification). Some more elaborated models also account for interactivity (e.g. [31]). Others concentrate on logical structure and navigational concepts for hypermedia (e.g. [33, 51]). Another problem with many of these models is that they are not intended to be directly used by a multimedia software engineer during a development project. Instead, they form the conceptual basis of (proprietary) authoring systems. Thus, they are not suited for directly supporting modeling activities in a multimedia software development process.

In traditional software engineering, the Unified Modeling Language (UML [47, 11])) has become the *de facto* standard notation for software development from the early stages of requirements specification up to detailed design. It has been adopted as the standard modeling language by the Object Management Group (OMG) and has been submitted for standardization to the International Standardization Organization (ISO).

UML is a family of visual, diagrammatic modeling languages that facilitate the specification of both software systems and process models for their development. The diverse language elements of its constituent sub-languages enable modeling of all relevant aspects of a software system, and methods and pragmatics can be defined how these aspects can be consistently integrated.

Unfortunately, UML does not support all aspects of multimedia applications in an adequate and intuitive manner (see Sect. 4.1). Especially, language features for modeling time, synchronization, and user interface aspects are not explicitly provided. Other concepts of UML are not mature enough or less vivid and thus aggravate multimedia modeling unnecessarily.

But UML comes with built-in extension mechanisms and a so-called profiling mechanism [47] in order to adapt and extend the general-purpose modeling language for specific development processes and application domains.

These fundamental concepts of the UML form the basis for the object-oriented modeling approach for multimedia applications that is presented as the main content of this chapter. Based on the characteristics of multimedia applications, we have developed extensions of the UML to specify all relevant aspects of multimedia applications in a single and coherent model.

The aim of this contribution is therefore to give an overview of existing approaches to object-oriented multimedia software modeling and, in particular, to introduce the UML-based, visual multimedia modeling language OMMMA-L (**O**bject-oriented **M**odeling of **M**ulti**M**edia **A**pplications - the **L**anguage). In this intention, we first identify the characteristic dimensions of multimedia software engineering and their relevance for an object-oriented modeling approach for multimedia applications in Sect. 2. We then give an overview of the state of the art in (object-oriented) multimedia software development. The UML-based modeling approach OMMMA is presented in Sect. 4 where we point out the essential elements of OMMMA-L for the specification of the different aspects of multimedia applications. Section 5 sketches some future directions of research and practice in the area of multimedia software engineering influenced by object-oriented modeling, and Sect. 6 concludes this chapter by summarizing the current achievements.

2 Dimensions of Multimedia Software Engineering

Software engineering can be approached from both the process and the system perspective. Within these perspectives, a wide range of dimensions exist that need to be considered. Each dimension captures a particular aspect of software engineering. But these dimensions do not exist in isolation, they have interdependecies with each other. Dimensions can be represented by partial models of software engineering that must be integrated. A formal, graph-based model for integrated software engineering has been presented in the GRIDS approach by Zamperoni (see [63]). The instantiation of this generic (meta-) model is exemplified by a three-dimensional model of software engineering 3D-M. This model identifies software processes, architectures, and views of the system as the three fundamental dimensions of software engineering and addresses the integration of the corresponding partial models.

The *process* dimension distinguishes different development phases (or activities) such as requirements specification, analysis, design, implementation, and testing. Modeling and implementation phases generally correlate to different levels of abstraction. Coordinates of the *architecture* dimension are different system components, e.g. interfaces with the user or external systems, control, processes, or a repository, whereas the *views* on the system can be distinguished in e.g. structure, function, dynamic behavior etc. Each view captures specific aspects of the system. (Note that architecture can also be regarded as a high-level internal view of the system to be built.) Software development can thus be understood as a (multi-) path through the multi-dimensional space of this integrated software engineering model, where the trajectory is defined by a process model, and suited software engineering technologies on the levels of concepts, languages, methods, and tools are applied.

If we look at the characteristics of multimedia applications and their development processes, we easily observe the necessity to tailor and extend these general dimensions. Multimedia specific aspects introduce new process activities, new requirements regarding the architectural structure, and new views on the system.

The *process* dimension must be extended to account for specific life-cycles of multimedia applications (see e.g. [52]) and to include activities of content and media production. Traditional development activities must be adapted, e.g. through new kinds of testing.

The *architecture* dimension needs to be refined for multimedia systems. *Multimedia systems* require complex architectures combining a multitude of hardware and software components. Thus, architectural design resembles software-hardware co-design of embedded systems. Architectural extensions are needed to account for different notions of media that co-exist in a multimedia system such as perception, presentation, representation, storage, and transport media (see [36]), or for media-related processors like filter, converter, and renderer components. Software components of multimedia systems can be categorized into system, service, and application components. *Multimedia applications* are interactive and have a strong emphasis on a multimedia user interface. Thus, software architectures from the field of interactive system modeling, e.g. Model-View-Controller (MVC [39]) or Presentation-Abstraction-Control (PAC [16]) can be considered as a basis for the architecture of multimedia applications. (A similar approach was followed in the development of the MET++ framework [1].) Since multimedia applications in general allow for multimodal interaction, architectures for interactive systems have to be specialized by adapting the control component to incorporate support for modes of interaction.

Views need to be added or reconsidered: the structure view has to account not only for domain objects, but also for media objects of static and temporal media types. Multimedia-specific application frameworks and patterns may be deployed. A presentation view, i.e., the design of the user interface with respect to spatial (and temporal) relations of its constituents becomes fundamental – more than in traditional interactive systems. Although space and time are mainly perceivable at the user interface, other system entities, such as media or domain entities, are subject to spatial and temporal relations (or constraints), too. In fact, the most important new aspect of multimedia systems is that of space and time, i.e., whether parts of the system have a temporal expansion relative to some time axis or a spatial within some two or three-dimensional coordinate space. Thus, a temporal and a spatial view or an integrated spatio-temporal view are mandatory. Temporal behavior must also account for real-time features and synchronization. Behavior can be predefined (algorithmic) or interactive, i.e., non-deterministic at specification time, because the system has to react to (run-time) events that are unpredictable in time. Multimedia-specific events, e.g. regarding the processing of temporal media data or specific media system services, need to be incorporated.

As has been stated above, the dimensions for architecture and views on the system are interrelated with each other. The relations between different views are partly influenced by architectural structures of the system and, vice versa, views on particular application aspects influence the architectural separation of concerns. For example, the interactive behavior view has to account for degree and modes of interaction (encapsulated in user interface components). Due to multimodal inputs resulting in the same application events, the coupling of interactive control behavior and system function needs to be revised. The coupling of these views can, for example, be accomplished by the architectural separation of physical user actions in user interface components – wrapping input devices – from application behavior in process components, mediated by different levels of technical and logical events exchanged and handled by the respective architectural components.

In case of hypermedia systems, navigation views can be understood as a further refinement of both the structure and the function views. Other characteristics that can be represented by dedicated views are security, media presentation adaption, quality of service (QoS), or presentation environment (compare [55]).

Since the primary focus of this contribution is not on the multimedia software process dimension, but on the multi-dimensional modeling of multimedia applications, the dimensions of the system perspective, i.e., architecture and views, are mainly relevant in the following. What remains as a requirement for a modeling language from the process dimension is the necessity to support models with different levels of abstraction. From the aforementioned views, the following can be identified as being fundamental (see [50] for a discussion):

- media and application *structure*,
- spatio-temporal *presentation*,
- *temporal behavior* (function),
- interactive *control*.

Together with the architecture dimension, these views will be used in the remainder to characterize the different presented approaches.

3 State of the Art in Multimedia Software Development

Researchers and practitioners agree that multimedia systems pose new requirements on software engineering (see e.g. [12] or [45] for an overview of topics).

Research and development projects in the multimedia field have been mainly focused on either multimedia enabling system technologies and services, e.g. networking or multimedia database systems, or the use of scripting-based authoring systems as well as XML-based or object-oriented programming technologies for the implementation of multimedia applications. Multimedia software engineering in the sense of specialized multimedia software development process models and usable multimedia specification languages and methods has only drawn minor attention.

In recent time, three different kinds of approaching multimedia software programming have become dominant.

- Most multimedia applications and documents are nowadays developed using multimedia authoring systems. Multimedia authoring systems are visual, interactive programming tools for the development of multimedia applications that can be used by developers with different degrees of expertise.
- With the advent of more complex multimedia software that needs to be integrated with other sophisticated application features, frameworks and application programming interfaces complementing standard (object-oriented) programming languages or system software were introduced.
- With the growing importance of web-based multimedia applications that are portable between multiple platforms, XML-based approaches to multimedia document authoring like the Synchronized Multimedia Integration Language (SMIL [62]; see [32]) – uptaking the SGML-based HyTime

approach – are being implemented aside from proprietary web-enabled formats like Flash (by Macromedia) or Apple's Quicktime.

Since the focus of this chapter is on object-oriented modeling of multimedia applications, we refer to these implementation technologies only in their relations to object-orientation and/or modeling in the following. In particular, authoring systems are discussed in the perspective of integration with modeling, and object-oriented frameworks are presented that show fundamental concepts based on which object-oriented modeling can be easier understood. Nevertheless, we stress that implementation of object-oriented models of multimedia applications can be done with any implementation technology, regardless of its degree of object-orientation. The architecture and views of object-oriented models can, for example, be easily mapped to the main concepts of most authoring systems. Technology-specific decisions would only influence the modeling on the level of detailed architecture and design.

3.1 Multimedia Authoring Systems and Modeling

Authoring systems are (partly) visual programming environments that support *ad hoc* implementation and rapid prototyping of multimedia applications based on direct-manipulative graphical user interfaces and intuitive media production and processing metaphors. They are in general supplemented by a scripting language containing simple language constructs to program extended functionality by hand. Pre-existing media and user interface objects are coupled with built-in mechanisms for dynamic execution, e.g. event handling. Extensibility of this functionality is rather limited.

Architecturally, authoring systems integrate four main functional components: media object tools, composition and structure tools, interpreters, and generators. Multimedia authoring systems can be classified according to the media metaphors they deploy as their main abstractions for design and content provision:

- *Screen or card-based* authoring systems, e.g. HyperCard [2] or ToolBook [13], place media objects on cards, slides, or pages, and navigational interaction allows users to switch between these cards.
- In *icon or flowchart-based* systems like Authorware [42], media objects have iconic representations that are used as nodes in a navigational flow graph.
- Macromedia Director [43] is an example for a *timeline-based* authoring system where media objects are positioned along a time axis and navigational interactions lead to jumps on this axis.

Regarding behavior, both card- and timeline-based categories rely in general on scripting-based event handlers that are associated with user interface elements, while dialogues in flowchart-based systems can be graphically specified.

Authoring systems are object-based rather than object-oriented, i.e., developers work with objects and component instances instead of classes and component types. Reuse in the form of composition of existing artifacts is possible only on the level of instances and scripts specifying object behavior. Inheritance is, if at all supported, restricted to the instance level.

Other disadvantages of most authoring systems are scripting languages built from primitive programming constructs; weak support for structuring, modularization, and reuse, especially in timeline-based systems; unsufficient support for team development; lack of user-defined types; limited documentation generation; and limited semantic foundation for analysis, test design, and maintenance. A striking disadvantage of most authoring systems is that they do not offer open and standardized interfaces for individual extensions or adaptions. Furthermore, the authored applications generally are not platform-independent bcause they use proprietary formats and languages.

The wide use of authoring systems and the lack of sophisticated and practically approved multimedia software process technology are responsible for a multimedia development practice that is *de facto* truncated to implementation and testing phases. This leads to problems well-known in traditional software engineering like missing conceptualization and documentation. Although the multimedia software engineering process is not the topic of this chapter, we will shortly reference two approaches for an integration of UML-based modeling and tool-supported authoring. We hereby intend to show how the current practice of development can be maintained and extended by modeling activities.

Boles et al. deploy UML for modeling and Director [43] for implementing a virtual genetics lab (see [10]). They use class, sequence, and statechart diagrams of UML to specify the application structure and behavior, i.e., possible user interactions as well as internal course of control, of simulated lab experiments. The Model-View-Controller (MVC [39]) paradigm is used as a pattern for implementation, it is not explicitly represented in the model. To transform the model to an executable application, they propose a translation approach into Lingo, the scripting language of Director, bypassing the visual programming capabilities of the authoring system. Events that trigger transitions in statechart diagrams are implemented as methods of the corresponding classes. Afterwards, view and controller classes for the direct-manipulative user interface are added. In this case, modeling is restricted to the application internals, user interface aspects are instantly coded.

To overcome the limitations regarding software engineering principles, we have proposed a process model for improving the development of multimedia applications from a software engineering perspective in [17]. It combines object-oriented modeling in analysis and design phases with an implementation based on a commercial authoring system, exemplified with Director [43]. The main idea is to transform a framework-based analysis model of the application, that is independent of the technology used for implementation, into a program. Key feature of this transformation is a conceptual programming model of the authoring system that bridges the gap between analysis model and implementation. This authoring system model is used during design to map an object model instantiating the analysis-level class model to an object model instantiating the authoring system model. The resulting design-level instance model can then be implemented in a straightforward manner. This approach has so far only covered the structural aspects of the multimedia application, it has not yet integrated a behavioral or dynamic view.

Depending on their main metaphors, authoring systems more or less explicitly support the different views on an application identified in Sect. 2. While the presentation view is generally well supported, complex application structures and behavior require sophisticated programming. The underlying architecture is transparent to the developer. Since authoring systems are visual programming environments, different levels of abstraction are not supported. From the above discussion we conclude that both multimedia authoring and object-oriented modeling can benefit from each other when they are used as complementary techniques within a multimedia software development process. Which role object-orientation plays in the current practice of multimedia software development will be presented in the next sections.

3.2 Object-orientation in the Development of Multimedia Applications

The important role of object-orientation for multimedia has been stated by many research contributions proposing object-oriented models as a conceptual basis for multimedia. Also, standardization efforts in the multimedia domain have discovered the advantages of object models. The family of MHEG standards [36] for the specifiaction of interoperable interactive multimedia documents (see [19] for an overview of MHEG-5 and its complementary parts) is based on an object-oriented model with abstractions for applications and scenes as well as links, streams, and other basic elements (so-called ingredients). Its focus is on coding and exchange of hypermedia documents. PREMO (Presentation Environment for Multimedia Objects [34]) is directed towards a standardization of the presentation aspects of multimedia applications. It incorporates temporal and event-based synchronization objects (see e.g. [29, 30, 28]). Its object model originates from the OMG object model for distributed objects. PREMO contains an abstract component for modeling, presentation, and interaction that combines media control with aspects of modeling and geometry. For the definition of the MPEG-4 [35] standard, objects have been discovered as a potential source for compression of video data instead of data reduction based on image properties.

But object-orientation is also promoted by the existence of object-oriented class libraries, toolkits, and frameworks that support the programming of multimedia applications. In the following, we summarize some characteristics of the latter since some of these approaches are accompanied by graphic authoring systems that enable visual programming based on an object-oriented conceptual model.

3.3 Object-oriented Multimedia Frameworks

Besides authoring systems, several object-oriented toolkits and frameworks have been proposed to support a programming-based development of multimedia applications. They reify architectural structures and fundamental abstractions of multimedia systems based either on extensions to object-oriented programming languages or on conceptual object-oriented languages that abstract from concrete programming environments. An important characteristic feature of object-oriented multimedia frameworks is their open and extensible architecture. It supports portability and adaptability, e.g. in the advent of new media types that can be integrated. Some developments in this field are, beyond others, the Media Editor Toolkit (MET++ [1]), MultiMedia Extensions (MME [18]), the Berkeley Continuous Media Toolkit (CMT [54]), and Nsync [4]. A recent development is the Java Media API [57] consisting of several components such as the Java 2D and 3D APIs or the Java Media Framework (JMF [58]; see e.g. [25]) for continuous media. Microsoft's DirectX [44] is a multimedia extension on the operating system level.

The main objective of object-oriented multimedia frameworks is to supply a multimedia developer with a software abstraction for multimedia programming. The framework should comply with the fundamental object types and operations that appear in multimedia applications. Conceptually, a framework consists of interrelated abstract classes that have to be implemented by concrete classes for different multimedia platforms. Therefore, one can distinguish at least two layers within such a framework. On the higher level, the abstract framework classes build an application programming interface that can be used by a multimedia developer independently of the target platform when implementing a multimedia application. On the lower level, concrete classes realize a platform-dependent implementation of the abstract concepts on the higer level. To achieve such an implementation, the framework classes make use of a system programming interface. The framework classes are organized in a generalization hierarchy where the abstract superclasses specify interfaces that are realized by their specialized, concrete subclasses.

Requirements for a multimedia framework are openess, robustness, ability to be queried, scalability, support for architectural structuring, availability of general, high-level concepts and interfaces for spatio-temporal media composition and synchronization, hardware control, database integration, and concurrency.

We will shortly refer to some representative frameworks in the following.

Framework by Gibbs and Tsichritzis. A prototypical object-oriented multimedia framework has been presented by Gibbs and Tsichritzis [24]. It specifies an abstract application programming interface (API) that serves as a homogeneous interface to heterogeneous platforms. It combines two fundamental concepts, a media hierarchy that encapsulates media values and a hierarchy of components. Transform and format class hierarchies are used as supporting concepts.

IMD. Another prominent approach that has been widely recognized is the modeling method for interactive multimedia documents (IMD) by Vazirgiannis [59]. It integrates the temporal and the spatial domain of multimedia documents in a common event-based framework. It comprises an object-oriented event model where elementary events of different, hierarchically specialized event types are combined by algebraic and spatio-temporal operators. Events are conceived as representatives of actions that generate these events, e.g. start or end of an action, parameterized by their subject (triggering object) and object (reactive object) and a spatio-temporal signature. Composite objects are combined from basic media objects by temporal and spatial operators. Based on these concepts, an authoring systeml is provided that allows developers to specify scenarios as a set of autonomous functions, so-called scenario tuples, to which start and end events, action lists and synchronization events, raised at the begin or end of the tuple, can be assigned.

MET++. MET++ [1] is an application framework in that multimedia presentations are modeled as hierarchical compositions of temporal objects. The modeled compositions are automatically transformed in temporal layout mechanisms and propagated. Real-time behavior and user interaction are integrated in the controller part of an extended MVC model. The MET++ class hierarchy includes compositional time-layout objects providing synchronization behavior for temporal relations and time-dependent media objects. Dynamic behavior of temporal media objects is specified by time-related functions that are themselves specializations of complex temporal objects.

Java Media API. The Java Media API [57] contains classes for the integration of animation, imaging, two and three-dimensional graphics, speech, and telephony. The Java Media Framework (JMF [58]), which is part of the Java Media API, offers an interface for accessing and controlling continuous media objects. It does not come with a general time concept that would enable the integrated synchronization of temporal and static media, only synchronization of the former is supported. Furthermore, no sophisticated mechanism for temporal composition is built in.

The presented frameworks and their inherent structuring can be used as sources for the architectural structuring of multimedia systems and applications, and partly for modeling language design. Additionally, their implementations are promising technologies for the implementation of object-oriented models of multimedia applications, especially for complex applications. But all the different views on multimedia applications must be mapped to basic object-oriented programming principles such as objects, messages, and events. Only in cases where the frameworks are themselves accompanied by graphical development (authoring) tools, like MET++, different views are explicitly supported on a higher (visual) level of abstraction.

We now step from object-oriented implementation technologies to modeling techniques. We first direct our attention to object-oriented modeling approaches for hypermedia and interactive systems that can contribute to a holistic multimedia modeling, before we return to the object-oriented modeling of multimedia applications in Sect. 4.

3.4 Hypermedia Modeling

Hypermedia software development has been addressed by different modeling approaches. Because these models focus mainly on *hyperlinked* media, emphasis is put on the design of navigational structures. Conceptual models of a hypermedia application are accompanied by some form of navigation model and sometimes by an abstract user interface model. Other aspects of multimedia, especially temporal behavior of continuous media and synchronization, are underrepresented.

The Object-Oriented Hypermedia Design Model (OOHDM [51]) starts modeling with a conceptual model of the semantics for the application domain that is complemented by a navigational model in a second step. This part of the model is based on an extended Entity-Relationship model. As a third activity, abstract user interface design completes the model. The browsing semantics of OOHDM is based on the static navigation structure specified in the navigational model. OOHDM is an object-oriented extension of HDM [23] and comprises the same basic modeling activities.

The Relationship Management Methodology (RMM [33]) is a method for design and implementation of hypermedia systems. In contrast to OOHDM, navigational structures are modeled within the domain model. Furthermore, RMM enables the generation of HTML documents or code for authoring systems from the model.

HyDev [48] is another proposal for decomposing hypermedia application models in different partial models. The domain model is accompanied by an instance model whose objects are instances of the domain model. The instance model is regarded important since the behavior of multimedia applications often relies on the characteristics of individual objects, and thus multimedia software development has to deal with both type and instance level views. Relevant features of object presentation and navigation between objects are captured in a so-called representation model that abstracts from media formats and concrete media objects.

HyperProp [55] comprises a conceptual model that represents authoring requirements by spatio-temporal constraints, and a formatting algorithm for runtime presentation adaption in reaction to occurrences of specified events. HyperProp is based on a logical document model for the composition of hypermedia artifacts. Architecturally, it distinguishes three layers for representation objects, data objects, and storage objects, respectively. The authoring system prototype contains a graphical editor for constraint specification.

From the perspective of modeling multimedia applications by extensions of the Unified Modeling Language, the work by Baumeister, Hennicker, Koch, and

Mandel is of particular interest. In [5], they propose extensions of the UML to specify hypermedia based on the concepts of OOHDM. In [27], these extensions are used for a further underpinning of the process associated with their language extensions. There, they give a set of guidelines how to (semi-) automatically derive information for a model view from previous models. They start with a conceptual model of the application domain from which they derive a navigation space model for the hypermedia application based on views on conceptual classes and additional navigation associations. From the navigation space model, they build a navigational structure model by incorporating navigational elements such as index, guided tour, query, and menu. Finally, they use the navigational structure model to construct an abstract presentation model focusing on the structural organization of the presentation rather than on the physical presentation on the user interface. The disadvantage of this approach is that interaction is restricted to the navigation via predefined links within the application.

In summary, the main shortcoming of hypermedia approaches is their limited capability of modeling behavior that is in most cases restricted to hyperlink navigation. Other forms of user-initiated control and temporal behavior are only partly considered. Additionally, application structure is often restricted to trees with hyperlinked nodes. Because multimedia applications are highly user-interactive, we look at user interface modeling next.

3.5 User Interface Modeling based on UML

In the UML field, there have also been some research contributions on how to extend the general purpose modeling language UML towards a better representation of user interface modeling dimensions. These are important for multimedia modeling since we have identified the multimedia user interface presentation and interaction as key views of multimedia systems in Sect. 2. Human-computer interaction has to deal with representations of user roles; of user behavior when performing tasks; of abstract conceptual and concrete physical user interfaces. To capture user roles and user requirements, use case diagrams of UML can be deployed. This approach is described in detail by Constantine and Lockwood [15]. In [60], so-called user interaction diagrams (UID) are introduced to detail use cases for requirements specification. For the modeling of behavior from the user's perspective, Kovacevic [38] proposes a UML extension for task analysis.

From a software analysis and design perspective, the UML profile for interaction design suggested by Nunes and Cunha [46] is of importance. Here, analysis and design models for the specification of user interfaces are presented. The analysis classes from the UML built-in profile for software development processes [47] are further refined to architecturally distinguish between interfaces to external systems and for user interaction. On the design level, a dialogue model for structuring dialogues and a presentation model for capturing navigation between different interaction spaces (contexts) are introduced.

The problem with all these models is that they are either on a high level of abstraction such as use cases or other requirement gathering approaches or they mostly focus on the architectural dimension or structural rather than behavioral aspects (views). An exception is the UML-based approach described in [53] that addresses dynamics of abstract user interfaces. Extended UML activity diagrams are employed to detail the interaction for realizing a use case. But since activity diagrams are still rather high-level behavior descriptions, they are not well-suited to describe detailed behavior of a concrete user interface.

None of the presented approaches of user interface modeling based on UML accounts for specific characteristics of multimedia applications and their implications on user interface modeling or the integration of multimedia applications with the proposed user interface models. Multimedia-specific architectures or system views beyond interaction are hardly supported. In the following section, we show how the OMMMA approach attempts to integratedly specifiy the different aspects of multimedia applications that have been identified in Sect. 2.

4 OMMMA — Object-oriented Modeling of Multimedia Applications

In this section, we introduce the modeling language OMMMA-L [50]. We show how this language extends the standard object-oriented modeling language UML appropriately and allows all aspects of a multimedia application to be modeled in an integral and coherent form.

4.1 UML and its Extensibility Towards Multimedia

The Unified Modeling Language (UML [47], see [11, 49] for an introduction, [21] for an overview) consists of a family of diagrammatic languages which are tailored to modeling diverse aspects of a system. Those are grouped into four categories: use case diagrams, structural diagrams, behavioral diagrams, and implementation diagrams. While use case diagrams are intended for capturing functional requirements of a system, implementation diagrams are used to describe physical system structures and runtime entities. Structural aspects are modeled in class and object diagrams. Behavioral aspects can be described using sequence, collaboration, statechart, and activity diagrams. For the modeling of multimedia applications, we have to analyze how well these diagram types are suited for modeling the architecture of multimedia applications and the four fundamental system views identified in Sect. 2: media and application structure, spatio-temporal presentation, temporal behavior (function), and interactive control.

Use case diagrams and implementation diagrams can be used to model requirements and architectural structure and components, respectively. In the following, we will focus on the fundamental system views within analysis and design models. Thus we only discuss the structural and behavioral diagrams regarding their appropriateness. As it turns out, the structure of an application can be adequately modeled in UML class (and object) diagrams, interactive control can be modeled in statechart diagrams, accompanied by a tailored dialogue signal hierarchy in class diagrams (although some specific abstractions for dialogue and user interaction specification as they are discussed in Sect. 3.5 may be desirable), and parts of (predefined) temporal behavior can be adequately modeled with UML sequence diagrams. But the analysis of UML's features reveals that specialized and more advanced language constructs are needed to describe the temporal assembling of different objects. Additionally, UML does not offer an explicit notation for spatial modeling in order to specify e.g. the presentation view of user interface layouts intuitively. Finally, UML lacks appropriate pragmatic guidelines on how to deploy the different diagram types cooperatively to model complex multimedia applications. Such guidelines relate to both which diagram types to use for a particular view on the system and how to deploy a particular diagram on a specific level of abstraction, and how the different views and levels of abstraction relate to each other. (Note that we concentrate on a single level of abstraction herein.) These shortcomings have led to the development of an extension of UML towards multimedia entitled OMMMA-L (**O**bject-oriented **M**odeling of **M**ulti**M**edia **A**pplications – the **L**anguage) that captures the application characteristics represented in the different views and to deriving pragmatics on how to model multimedia applications with an object-oriented language based on UML.

The extensions of OMMMA-L can be integrated with UML by deploying UML's built-in extension mechanisms allowing existing model elements to be specialized by stereotypes, constraints, and tagged values (see [47]). These lightweight extensions do not influence the syntax and semantics of the UML itself, but semantics can be specialized for domain-specific extensions. The extensions can then be used to build profiles for specific application domains or kinds of applications.

OMMMA-L is presented in the next subsections, starting by introducing an example application to be modeled.

4.2 OMMMA-L Modeling Example: Automotive Information System

The UML-based modeling language OMMMA-L has been designed to model a wide range of aspects of interactive multimedia applications. We illustrate its capabilities by showing extracts from a model of a simulation application of an automotive information system.

Car cockpits nowadays evolve towards being highly-integrated multimedia information interfaces that interact with many embedded components as well as with external and distributed information systems and services. Diverse applications have to be integrated, like car audio, navigation and communication systems, travel or tourist information, and automotive system monitoring and control.

Regarding interactivity with a human user, several levels of abstraction can be distinguished: on a low level of interaction, a user has to interact with hardware input and output devices (presentation media) that are visual, haptic, or voice-enabled. Input devices produce signals that need to be transformed to

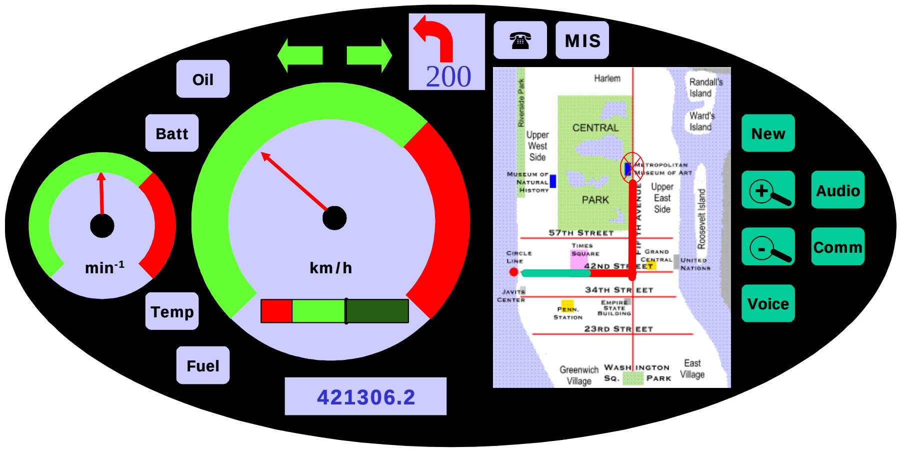


Figure 1: Display of an Automotive Information System

semantic events on application level. For examle, clicking the right mouse button on a specific point on the screen has a specific semantics for the application when it is in a particular state. Pressing a specific button in a multi-functional automotive control panel also shows a context-dependent behavior.

We return to the appropriate aspects of this application in the following subsections in order to illustrate the language concepts of OMMMA-L. The four fundamental views identified in Sect. 2 each relate to a particular diagram type in the OMMMA approach:

- media and application structure are modeled in the class diagram;
- the spatial aspect of the presentation is modeled in OMMMA-L presentation diagrams (that are related to OMMMA-L sequence diagrams for the spatio-temporal integration);
- temporal behavior (function) is modeled in OMMMA-L sequence diagrams; and
- interactive control is modeled in statechart diagrams.

Architectural considerations also appear on these diagrams, although they are not explicitly modeled within these OMMMA-L diagram types. The individual diagrams are presented in the succeeding sections, before we explain the integration of these diagrams in Sect. 4.7.

4.3 Class Diagram

Class diagrams are the core of an object-oriented application model and are used to model the static structure of the multimedia application. Essentially,

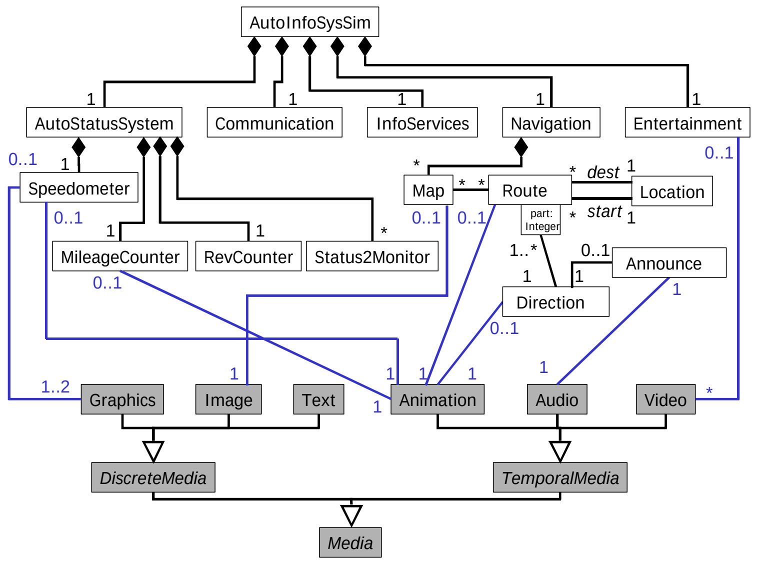


Figure 2: OMMMA-L class diagram

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 structural model aspects of application semantics and media types, each OMMMA-L class diagram consists of (at least) two closely interrelated parts:

- an *hierarchy of media type definitions*, which comprises classes for all (representation) 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. For the specification of interaction, these class hierarchies must be accompanied by a *signal hierarchy* as a basis for event-based interaction. *Presentation classes* may be deployed (possibly in a different package) in addition to model the possible composition of user interfaces as a basis for the presentation diagrams introduced in Sect. 4.5. Figure 2 shows a part of the class diagram for the sample application of a simulated automotive information system shown in Fig. 1. The lower part of the diagram depicts the media type hierarchy and the upper part the structure of the logical model. It shows that the automotive information system simulation is a complex composition of five subsystems: an automotive status system and systems for navigation, communication, information services, and entertainment. The status and navigation systems are more detailed. The status system comprises a speedometer, a mileage counter, a revolution counter, and a set of status monitors. The navigation system contains multiple maps that can be associated with an unrestricted number of routes. In turn, routes can be related to multiple maps. For each route, there is a start and a destination location. A route relates to a set of directions that are qualified from the perspective of a route by a part number defining their position in the sequence of directions. Additionally, a direction may be accompanied by a spoken announcement of the way to drive. For some of these application classes, the associations to elements of the media class hierarchy are shown. The speedometer, for instance, is related to one or two graphics and an animation, e.g. to enable a day and night design of the background and and a moving indicator for presentation of the actual speed. Also, routes shown on the map (which is related to an image) and the directions are realized as animations. An announcement is related to an audio object whereas the (simplified) entertainment system relates to multiple video objects.

OMMMA explicitly distinguishes between application objects as regards content and media objects in order to allow an application to present one application entity by different (representation) media, e.g. accounting for distinct presentation media, such as screens, or resource availability. Thus, media objects are not specializations of application objects or vice versa. The dimension of media types is based on a generalization hierarchy of static and temporal media types as it can be found in several multimedia standards, e.g. MHEG [36], and frameworks, e.g. the framework by Gibbs and Tsichritzis [24] or MET++ [1].

4.4 Extended Sequence Diagram

UML offers various diagram types to model behavioral aspects of an application. Due to their emphasis on modeling temporal sequences (of messages), 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 are extended by a series of features, especially regarding timing and time constraints. These are for example:

- *Refinement of the time dimension* by defining local time axes for objects supporting a notion of local time. Local time can be related to global (real) time (represented by the actor's timeline) to specify *intra-object* synchronization or to the time of other objects to specify *inter-object* synchronization. Durations and points in time can be specified by different forms of fixed, bounded or unbounded time intervals restricting their possible temporal positions. Time intervals are represented by their start and end points. (Syntactically, these timing requirements can be written as constraints using (in-)equalities or in an interval notation.)

- *Sychronization bars* (bold lines) instead of message arrows between object activations to specify the continuous inter-object synchronization between

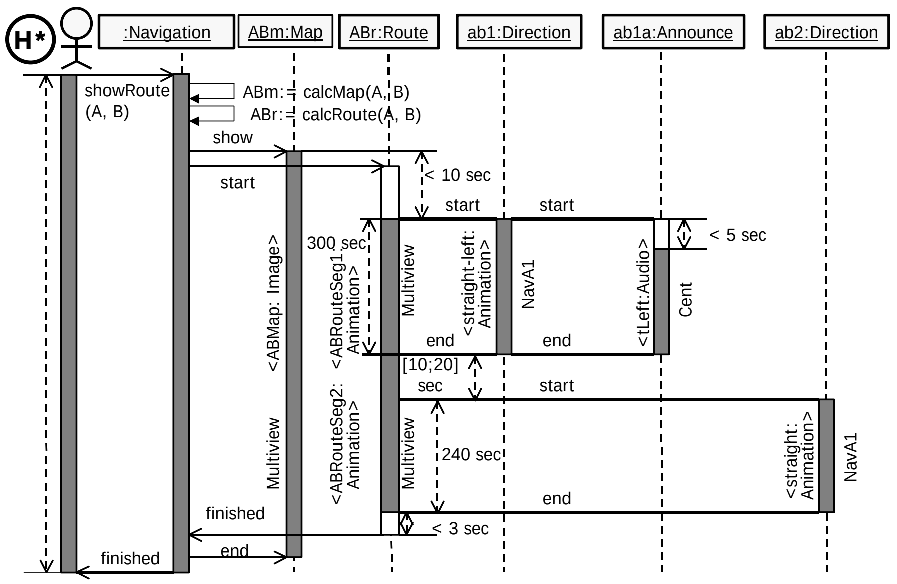


Figure 3: OMMMA-L sequence diagram

temporal media presentations that may abstract from a message direction.

- *Activation and deactivation delays* of media objects in order to model tolerated variations of synchronization relations for media objects (compare maximum start and end times in [31]).

- A notion of shallow and deep *History* on a sequence diagram that UML only provides for statechart diagrams. It allows the designer to specify whether a specified scenario can be interrupted and later returned to at the same point of virtual time where it had been interrupted. Deep history (H*) denotes that this temporal reconstruction is possible even on nested invocation levels with the semantics of pause and resume for the possibly complex presentation, whilst shallow history (H) restricts returning to the situation on the top level with the semantics that the sequence diagram can be restarted from the interruption time, but already started presentations of objects cannot be cued to their last presentation state.

- *Parallely composed activation* of media objects in order to model the simultaneous presentation of an application object across different presentation channels (or media) and/or by different media objects.

- *Sequentially composed activations* resulting in an *automatic triggering* of subsequent activations (segments), e.g. to present an animated object that is sequentially presented via different channels and/or by different media objects.

- Activations of application objects can be annotated with presentation objects, either abstractions from hardware devices (presentation media) or software user interface objects – depending on the level of use – such as audio channels or graphical objects on a screen, and/or media objects designated to represent an application object during an activation or activation segment. Associated media objects, that must conform to the types specified in the class diagram, are enclosed in ⟨⟩. The identifiers of presentation objects appear as pure strings (as they are used on presentation diagrams, cf. Sect. 4.5).

- Activations of objects may be overlayed by *media filters*, which describe time functions, e.g. the incremental increase of an audio level over time.

Each OMMMA-L sequence diagram models the temporal behavior of a predefined *scenario* of the multimedia application. The scenario specified by the sequence diagram is represented by the (initial) message sent from an actor symbol (or some user interface component) to an object within the sequence diagram that acts as the scenario controller. The message can be parameterized, e.g. by time stamps for start and end of execution of a sequence diagram, in order to support its re-use, or by parameters that may be used in guard expressions or nested message calls.

All objects in one diagram relate to the same (global) timeline to which they can be synchronized if required. The projection of a single continuous media object on the corresponding (global) timeline specifies intra-object synchronization. Concurrent scenarios with an independent timeline need to be modeled by different sequence diagrams related to parallel substates within an *and*-superstate of the corresponding statechart diagram (see Sect. 4.6), i.e., they do not have a common notion of time. Different message types between objects enable specification of synchronous or asynchronous messaging. A propagation mechanism has to ensure consistency of temporal specifications or the mapping of a local time axis to its relative temporal coordinate system.

Figure 3 gives an example of an OMMMA-L sequence diagram. It describes the execution of showRoute(Location A, Location B) which is an operation of the navigation system. All objects shown in the horizontal object dimension are semantic, i.e. application objects. Based on the parameters for start and destination (A respectively B), the Navigation object determines a route object ABr and a map object ABm that is then called to be shown. After a maximum activation delay of ten seconds relative to the start of the presentation of the map, an animation of the route from location A to B has to be presented. This animation consists of two parts of which the first one is associated with the media object ABRouteSeg1:Animation that is shown for 300 seconds and the second one is associated with the media object ABRouteSeg2:Animation that is shown for 250 up to 260 seconds. (These presentation time intervals do not necessarily coincide with the duration of the animation objects themselves.) Both animations are presented via a screen object referenced as Multiview (see Fig. 5). Parallel to the first part of the route animation, direction ab1:Direction is presented at NavA1. It is accompanied by a spoken announcement that must be started at most five seconds after the invocation by the direction object ab1, and whose end is synchronized with the ends of both the first part of the route animation and the corresponding direction ab1. The annoucement is to be output via a center speaker denoted by Cent as specified in Fig. 4. The synchronization bars at the start and the end of these activations specify that a continuous inter-object synchronization is intended and has to be ensured by the renderer at presentation time. The end of the first part of the route animation directly triggers the execution of the second part. After a delay that is between 10 and 20 seconds, another direction animation ab2 is started. It (synchronously) co-ends with the presentation end of the second part of the route animation. After finishing its presentation, the route object has to signal to the navigation component within 3 seconds that presentation is finished.

For the specification of activation and deactivation delays, we use the UML presentation option to distinguish periods of actual computing by shading the activation segments from periods where objects are activated, but do not own the focus of control of the associated thread, by plain activation segments [47].

4.5 Presentation Diagram

Class diagrams are used to model the media and application structure view in OMMMA-L, OMMMA-L sequence diagrams model the temporal behavior (function) view. Before we continue with interactive control in the next subsection, we first explain how the spatial 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 modeling this view, the new *presentation diagram* type is added to OMMMA-L. Presentation diagrams support an intuitive description of the layout, i.e., the spatial arrangement of *presentation objects* at the user interface. Spatial relationships (and constraints) can thus be graphically depicted. 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 presentation area of the user interface by boundig boxes for presentation objects (more precisely, these can be roles as in UML collaboration diagrams that can be substituted by conforming objects) that are to be presented. Bounding boxes show geometry and size characteristics and are positioned on a virtual area relative to some specified coordinate system. Presentation objects are distinguished into visualization objects and interaction objects. *Visualization objects* are passive objects that are used to present e.g. text, pictures, graphics, video, or animation objects. *Interaction objects* enable user interactions and may raise events in the running system. Examples are scroll or menu bars, buttons, input fields or a hypertext containing links. Bounding boxes for interaction objects are indicated by bold borders (like active objects are marked in UML). The visual layout specification is accompanied by an iconic representation of audio channels beside the visual presentation area.

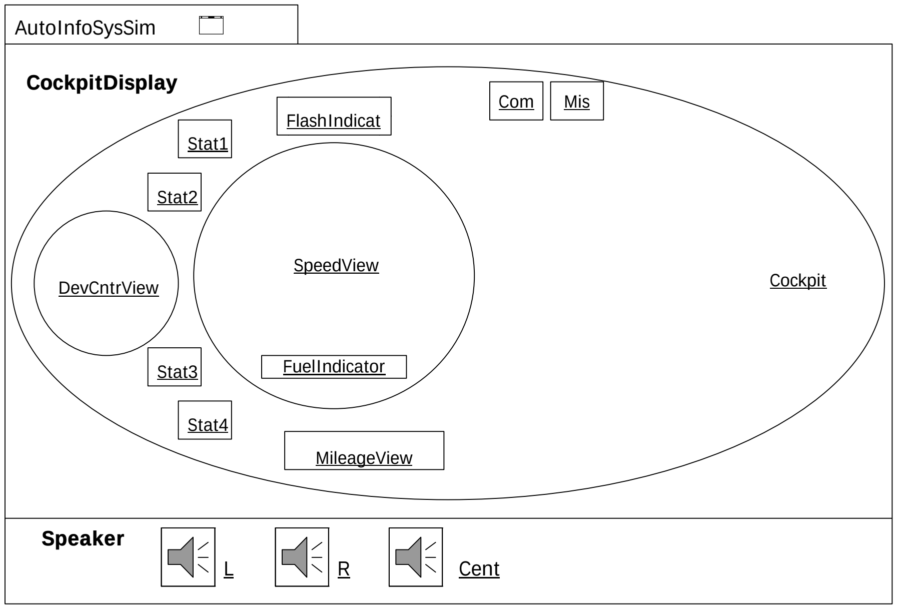


Figure 4: Application-level OMMMA-L presentation diagram

We use the notation of a stereotyped UML package to depict a presentation diagram. The presentation diagram can be further divided into different compartments representing hardware output devices such as different screens or audio channels. In Fig. 4, the presentation diagram is divided in two compartments, one for a dashboard cockpit display and one for the left, right, and center speakers as channels of an audio system (in the same way, local areas for input devices can be described).

Since, in our example, there is no direct interaction with the presentation objects like in direct-manipulative graphical user interfaces, but interaction is via specific input devices such as knobs, all bounding areas are marked as visualization objects (an area for input elements has been omitted).

The complete presentation of a certain application unit may be described by several presentation diagrams that may be composed by a layered placement on the virtual area (e.g. Figs. 4 and 5). Positioning of presentation elements is by convention relative to the directly surrounding element unless otherwise specified, e.g. by a separate hierarchical composition of presentation elements. For instance in Fig. 5, MultiView is positioned relative to Cockpit from the presentation diagram AutoInfoSysSim, given by the path expression on the diagram.

Following this description of spatial modeling of the presentation view, now the interactive control view, modeling system reaction to (user) inputs and other events, is shown.

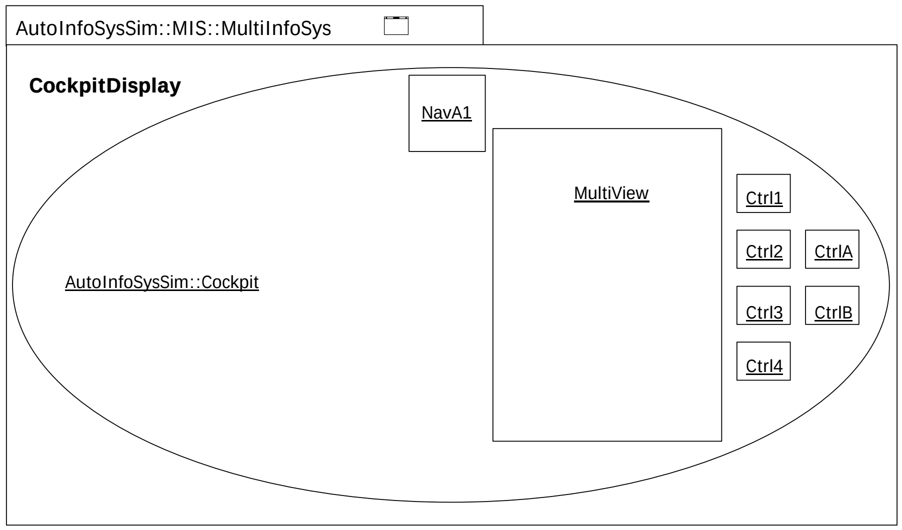


Figure 5: Lower-level OMMMA-L presentation diagram

4.6 Statechart Diagram

In UML, statecharts are assigned to classes to specify the behavior of their instances. They can be used on different levels of granularity. On a high level, they can specify the behavior of an application or substantial parts thereof. On a low level, they can be used to specify the behavior of simple classes, such as user interface element classes, to model e.g. the behavior of a button or the control state of a media object or its associated media player. The high-level statecharts are used to specify the control behavior within the context of the application since it must be specified which event regarding which particular element of the application triggers a transition from a state of the application. Examples of high-level statecharts that are partly schematic for simplification of presentation can be seen in Figs. 6 and 7. They are used on an application-semantic rather than a user interface level.

While OMMMA-L sequence diagrams are used to specify the (predefined) temporal behavior of a multimedia application, statechart diagrams are used to specify the system states as well as state transitions triggered by user interactions or other system events, i.e., the *interactive control* or dynamic behavior. OMMMA-L statechart diagrams are syntactically and semantically equal to UML statechart diagrams. This means that e.g. they may be structured by *and*- and *or*-superstates or refined by embedded statechart diagrams. An action appearing on an OMMMA-L statechart may represent a multimedia scenario (see Sect. 4.4). For example, internal entry- and exit-actions, or do-activities of states may be labeled with names of actions or action expressions.

To enable dialogue specification on an adequate level of abstraction, e.g.

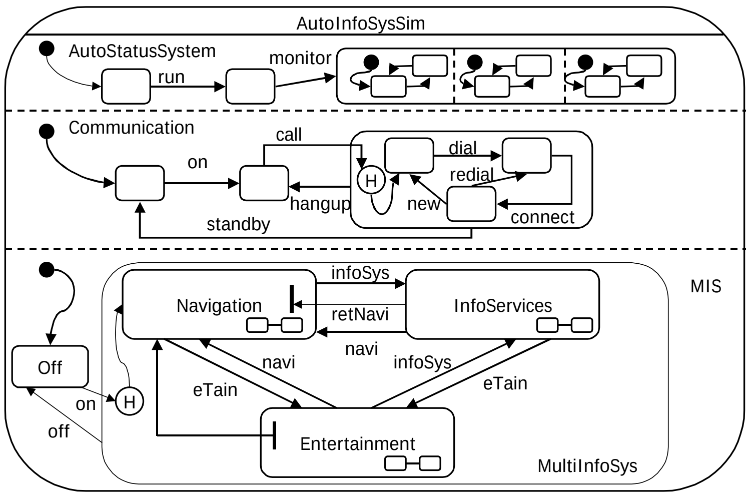


Figure 6: OMMMA-L statechart diagram for the automotive information system application

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 [47], cf. Sect. 4.3). 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. Events on statechart diagrams relate to such signals. Signals may be categorized according to user interaction events, application events, system events, and timer events. The event model that can be used within OMMMA-L statecharts is not restricted. It is therefore possible to integrate event algebras as they have been specified in the area of active database management systems, that have also been used as a basis for the spatio-temporal event model in [59]. Events can then be composed by algebraic and temporal operators such as and, or, or seq.

This description of the use of statechart diagrams in OMMMA-L concludes the presentation of the individual diagrams.

4.7 Integrated Modeling

Each of the above introduced OMMMA-L diagram types is used to specify a certain view of a multimedia application. What remains is the integration of these views into a coherent model of a multimedia application.

The typological fundament of the different views on the multimedia application is defined in the OMMMA-L class diagram. Other structural and behavioral views must conform to the class definitions and association specifications therein. This implies that objects on OMMMA-L sequence diagrams or – if intended – presentation objects on presentation diagrams must be typed

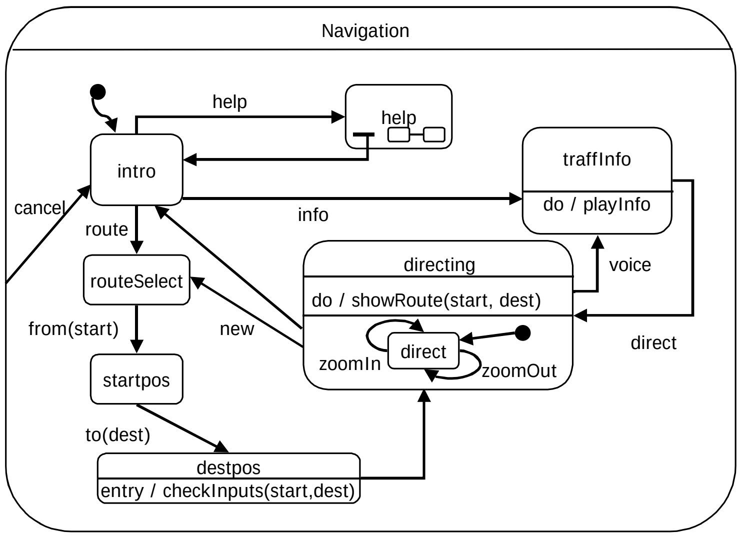


Figure 7: OMMMA-L statechart diagram for the navigation subsystem

over these classes. Statechart diagrams are assigned to these classes for the specification of interactive control or dynamic behavior.

A multimedia application may be more detailed seen as a collection of multimedia application units, so-called *scenarios* or scenes. In the OMMMA approach, each scenario corresponds to a state within an high-level statechart diagram which is associated to the class of the overall application or some class encapsulating a substantial part thereof. Furthermore, each scenario is via its associated state related to a presentation, possibly composed of different presentation diagrams.

A state associated to a scenario may be refined by a nested statechart diagram or nested states (as depicted for state directing in Fig. 7) which describe the possible interactive behavior during this scenario. (Thus, scenarios can be hierarchically composed.) For example, state directing is not left when zooming, implying the semantics that the temporal behavior of the scenario related to *showRoute* is not influenced by zoom operations.

In order to couple the interactive control with the predefined functional behavior views of a multimedia application, actions on a statechart diagram can be associated with OMMMA-L sequence diagrams that specify the scenarios, more precisely, the predefined, timed pieces of functional behavior within a scenario, corresponding to such actions or action expressions. An entry-action or do-activity means that the behavior specified by the sequence diagram is automatically triggered whenever the corresponding state is entered, exit-actions are executed before the state is left. The semantics of the (concurrently executed) do-activity is thereby to be interruptible by events triggering a transition from

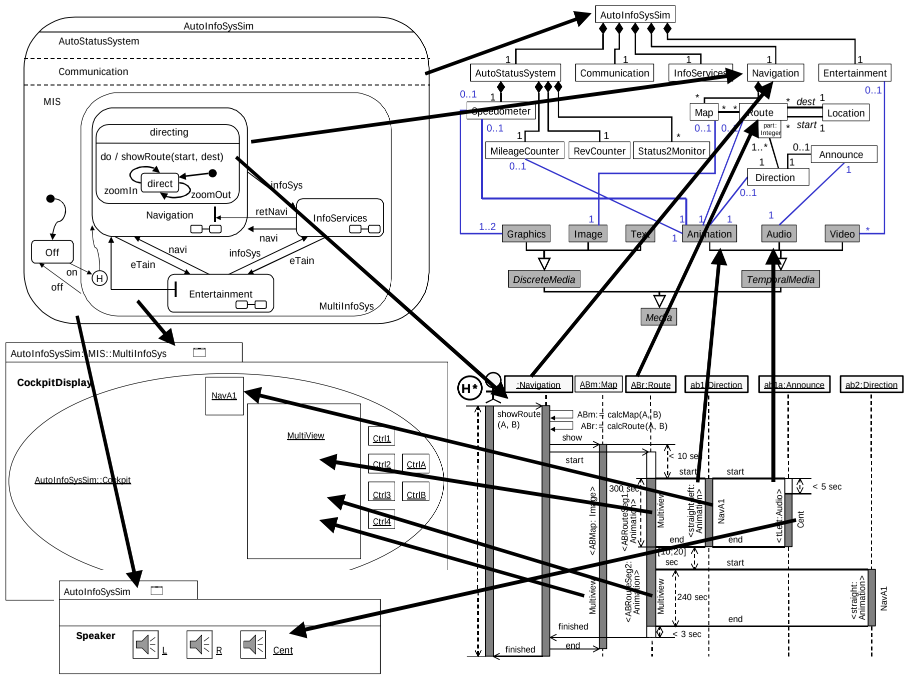


Figure 8: An integrated OMMMA-L specification

that particular state, whilst entry-actions and exit-actions are non-interruptible as specified by the UML run-to-completion semantics (see [47]). Figure 7 gives examples for an entry-action in state destpos and for two do-activities in states traffInfo and directing. The latter do-activity is specified by a sequence diagram with a (deep) history indicator denoting that the assigned scenario can be completely resumed after interruption. This construction is semantically feasible since do-activities on statechart diagrams are executed concurrently. Based on this coupling of function and control views, 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 [31].

A presentation diagram is associated to a state of the interactive control view in order to couple presentation and control views. Therefore, its name coincides with the name of the state to which it is assigned. Semantically, this presentation diagram will be part of the user interface presentation as long as the application or the respective part thereof is in that specific state. In Fig. 4, the presentation diagram is labeled with the name of the top-level state AutoInfSysSim, meaning that these objects are addressable for presentation as long as the application is running.

The composition of a complete presentation of a certain application unit (or state) is based on the hierarchical composition of states to which the constituent presentation diagrams are assigned. Figure 5 shows a presentation diagram that is assigned to state MultiInfoSys which is a nested substate of AutoInfoSysSim (via state MIS) as can be seen from the path expression in the name compartment in the upper left. When the application is in state AutoInfoSysSim::MIS::MultiInfoSys or a substate thereof, the complete presentation is composed of (at least the given) two presentation diagrams, i.e., the general diagram for all application states and the diagram for the multi-information system being active (compare Fig. 6).

In OMMMA-L sequence diagrams, an activation of an application object can reference a media object that is being presented during this activation as well as a presentation object that may be used on a reachable presentation diagram in order to specify the spatio-temporal constraints of the presentation to the user. Reachable presentation diagrams are defined by the coupling statechart diagram. They either relate to the same state to which the action belongs that is specified by the sequence diagram – either an internal action of that state or an action on a transition within that state in case of a complex state – or to a superstate of that state.

Figure 8 gives a simplified example of the interrelations between diagrams in 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 specified scenario (initial message) in a sequence diagram used as the action expression of an internal action within a state of the state diagram, or the name of a state used as the identifier of a presentation diagram, or the name of a (visual) presentation object or audio channel on a presentation diagram used within a sequence diagram in association to an activation box. Other relations (consistency constraints) between diagrams are depicted by overlayed arrows in Fig. 8. These constraints are precisely specified in a UML profile for multimedia applications that extends standard capabilities of UML for the modeling of multimedia applications according to the OMMMA approach.

5 Future Directions

The modeling approaches presented herein still have to show their feasibility in real-world applications and complex application settings. Case studies are necessary to underpin their real achievements in practice.

The extension of UML by profiles for specific application domains has been widely recognized as indicated by a diverse range of (prototypical) custom profiles, e.g. for architectures of web applications [14], and several requests for proposals issued by the OMG in order to standardize profiles for several domains, like embedded real-time applications and CORBA. A major drawback still is the absence of generally accepted, formal and precise specifications of UML semantics, and, therefore, also for most proposed extensions of UML. But with the forthcoming appearance of diverse profiles, the issues of consistency between and semantically sound integration of profiles need to be analyzed in detail. Orthogonal aspects should be placed in isolated profiles that could then more easily be combined. For the profiles presented herein, the integration of user interface, hypermedia, and multimedia modeling extensions is an interesting challenge.

A recent trend in the specification of multimedia applications is the use of constraint-based approaches to more flexibly describe the requirements and properties of multimedia presentations (see e.g. [40, 7, 56, 26]). Some constraints can be graphically expressed in the current OMMMA-L notation, other constraints can be integrated into a model by UML's built-in constraint language OCL [47] or by using other textual or diagrammatic (for instance, constraint diagrams [37]) constraint languages.

Some interesting challenges exist regarding the integration of the proposed object-oriented modeling approach into general multimedia software development processes. Especially the transformation between the model and an implementation, based on either object-oriented frameworks or authoring systems, that does account for both structure and behavior, is an obvious task at hand to prove the feasibility of the concept.

6 Conclusion

In this chapter, we presented approaches for the object-oriented modeling of interactive, hypermedia, and multimedia applications. We also examined the current state of application development based on multimedia authoring systems and object-oriented programming. By doing this, we illuminated some important aspects of the multi-dimensional task of multimedia software engineering.

In our presentation, we focused on 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 support integrated modeling of all aspects of a multimedia application. Particularly, a presentation diagram type and appropriate extensions to sequence diagrams have been introduced.

The modeling language is accompanied by a method description how to deploy the language elements in a multimedia software development process and a prototype implementation. Furthermore, the language extensions are being formalized by a precise semantics specification based on graphical operational semantics (cf. [20]). As a refined conceptual basis, we intend to define a formal model for the composition within and between the different behavioral diagrams.

References

[1] P. Ackermann, *Developing Object-Oriented Multimedia Software — Based on MET++ Application Framework* (dpunkt, Heidelberg, 1996).

[2] Apple, Hypercard, *http://www.apple.com/hypercard/*

[3] T. Arndt, "The evolving role of software engineering in the production of multimedia applications", *Proceedings of the IEEE International Conference on Multimedia Computing and Systems* (*ICMCS'99*) **I** (1999) 79–84.

[4] B. Bailey, J. A. Konstan, R. Cooley and M. Dejong, "Nsync — A toolkit for building interactive multimedia presentations", *Proceedings of the 6th ACM International Conference on Multimedia'98* (ACM Press, 1998) 257–266.

[5] H. Baumeister, N. Koch and L. Mandel, "Towards a UML extension for hypermedia design, eds. R. France and B. Rumpe, *Proceedings of the ≪UML≫'99 — The Unified Modeling Language, Beyond the Standard, 2nd International Conference*, *Lecture Notes in Computer Science 1723* (Springer, 1999) 614–629.

[6] E. Bertino and E. Ferrari, "Temporal synchronization models for multimedia data", *TKDE* **10**, no. 4 (1998) 612–631.

[7] E. Bertino, E. Ferrari and M. Stolf, "MPGS: An interactive tool for the specification and generation of multimedia presentations", *TKDE* **12**, no. 1 (2000) 102–125.

[8] A. Bianchi, P. Bottoni and P. Mussio, "Issues in design and implementation of multimedia software systems", *Proceedings of the IEEE International Conference on Multimedia Computing and Systems* (*ICMCS'99*) **I** (1999) 91–96.

[9] M. M. Blattner and E. P. Glinert, "Multimodal integration", *IEEE MultiMedia* **3**, no. 4 (1996) 14–24.

[10] D. Boles, P. Dawabi, M. Schlattmann, E. Boles, C. Trunk and F. Wigger, "Objektorientierte Multimedia-Softwareentwicklung: Vom UML-Modell zur Director-Anwendung am Beispiel virtueller naturwissenschaftlich-technischer Labore", *Proceedings of the Workshop Multimedia-Systeme, 28th Annual Conference of the German Computer Science Association* (*GI*) (1998) 33–51 (in German).

[11] G. Booch, J. Rumbaugh and I. Jacobsen, *The Unified Modeling Language User Guide* (Addison-Wesley, Reading, MA, 1998).

[12] S.-K. Chang, *Multimedia Software Engineering* (Kluwer, Boston, MA, 1999).

[13] Click2Learn, Toolbook II, *http://home.click2learn.com/*

[14] J. Conallen, *Building Web-Applications with UML* (Addison-Wesley, Reading, MA, 2000).

[15] L. L. Constantine and L. A. D. Lockwood, *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design* (ACM Press, New York, NY, 1999).

[16] J. Coutaz, "PAC-ing the architecture of your user interface", eds. M. D. Harrison and J. C. Torres, *Design, Specification and Verification of Interactive Systems, Proceedings of the 4th Eurographics Workshop '97* (Springer, 1997) 13–27.

[17] R. Depke, G. Engels, K. Mehner, S. Sauer and A. Wagner, "Ein Vorgehensmodell für die Multimedia-Entwicklung mit Autorensystemen", *Informatik: Forschung und Entwicklung* **14** (1999) 83–94 (in German).

[18] D. Dingeldein, "Modeling multimedia objects with MME", *Proceedings of the EUROGRAPHICS Workshop on Object-Oriented Graphics* (*EOOG'94*), 1994.

[19] M. Echiffre, C. Marchisio, P. Marchisio, P. Panicciari and S. Del Rossi, "MHEG-5 — Aims, concepts, and implementation issues", *IEEE MultiMedia* **5**, no. 1 (1998) 84–91.

[20] G. Engels, J. H. Hausmann, R. Heckel and S. Sauer, "Dynamic meta modeling: A graphical approach to the operational semantics of behavioral diagrams in UML", in [22], pp. 323–337.

[21] G. Engels, R. Heckel and S. Sauer, "UML — A universal modeling language?" eds. M. Nielsen and D. Simpson, *Proceedings of the 21st International Conference on Application and Theory of Petri Nets* (*ICATPN 2000*), *Lecture Notes in Computer Science 1825* (Springer, 2000) 24–38.

[22] A. Evans, S. Kent and B. Selic, *Proceedings of the ≪UML≫2000 — The Unified Modeling Language, Advancing the Standard, 3rd Intl. Conf.*, *Lecture Notes in Computer Science 1939* (Springer, 2000).

[23] F. Garzotto, P. Paolini and D. Schwabe, "HDM - A model-based approach to hypertext application design", *TOIS* **11**, no. 1 (1993) 1–26.

[24] S. J. Gibbs and D. C. Tsichritzis, *Multimedia Programming: Objects, Environments and Frameworks* (Addison-Wesley, Wokingham, 1995).

[25] R. Gordon and S. Talley, *Essential JMF: Java Media Framework*, (Prentice-Hall, Englewood Cliffs, NJ, 1999).

[26] V. Hakkoymaz, J. Kraft and G. Özsoyoglu, "Constraint-based automation of multimedia presentation assembly", *Multimedia Systems* **7**, no. 6 (1999) 500–518.

[27] R. Hennicker and N. Koch, "A UML-based methodology for hypermedia design", in [22], pp. 410–424.

[28] I. Herman, N. Correia, D. A. Duce, D. J. Duke, G. J. Reynolds and J. van Loo, "A standard model for multimedia synchronization: PREMO synchronization objects", *Multimedia Systems* **6**, no. 2 (1998) 88–101.

[29] I. Herman, G. J. Reynolds and J. van Loo, "PREMO: An emerging standard for multimedia presentation — Part I: Overview and framework", *IEEE MultiMedia* **3**, no. 3 (1996) 83–89.

[30] I. Herman, G. J. Reynolds and J. van Loo, "PREMO: An emerging standard for multimedia presentation — Part II: Specification and applications", *IEEE Multimedia*, **3**, no. 4 (1996) 72–75.

[31] N. Hirzalla, B. Falchuk and A. Karmouch, "A temporal model for interactive multimedia scenarios", *IEEE MultiMedia* **2**, no. 3 (1995) 24–31.

[32] P. Hoschka, "An introduction to the synchronized multimedia integration language", *IEEE MultiMedia* **5**, no. 4 (1998) 84–88.

[33] T. Isakowitz, E. Stohr and P. Balasubramanian "RMM: A methodology for structured hypermedia design", *CACM* **38**, no. 8 (1995) 34–44.

[34] ISO/IEC JTC1/SC24/WG6 — Multimedia Presentation and Interchange, PREMO (ISO/IEC 14478), *http://www.iso.ch*

[35] ISO/IEC JTC1/SC29/WG11 — Moving Picture Experts Group (MPEG), MPEG-4 (ISO/IEC 14496), *http://www.iso.ch*, *http://www.cselt.it/mpeg/*

[36] ISO/IEC JTC1/SC29/WG11 — Multimedia and Hypermedia Expert Group, MHEG (ISO/IEC 13522), *http://www.iso.ch*

[37] S. Kent, "Constraint diagrams: Visualising invariants in OO modelling", *Proceedings of the OOPSLA'97* (ACM Press, 1997) 327–341.

[38] S. Kovacevic, "UML and user interface modeling", eds. J. Bézivin and P.-A. Muller, *Proceedings of the ≪UML≫'98 — The Unified Modeling Language, Beyond the Notation, 1st International Workshop*, *Lecture Notes in Computer Science 1618* (Springer, 1998) 253–266.

[39] G. E. Krasner and S. T. Pope, "A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80", *Journal of Object-Oriented Programming* **1**, no. 3 (August/September 1988) 26–49.

[40] Y.-M. Kwon, E. Ferrari and E. Bertino, "Modeling spatio-temporal constraints for multimedia objects", *Data and Knowledge Engineering* **30**, no. 3 (1999) 217–238.

[41] T. D. C. Little and A. Ghafoor, "Synchronisation and storage models for multimedia objects", *IEEE Journal on Selected Areas in Communications* **8**, no. 3 (April 1990) 413–427.

[42] Macromedia, Authorware,
http://www.macromedia.com/software/authorware/

[43] Macromedia, Director, *http://www.macromedia.com/software/director/*

[44] Microsoft, DirectX, *http://www.microsoft.com/directx/*

[45] M. Mühlhäuser, "Issues in multimedia software development", *Proceedings of the International Workshop on Multimedia Software Development* (IEEE Computer Society, 1996) 2–9.

[46] N. J. Nunes and J. F. e Cunha, "Towards a UML profile for interaction design: The Wisdom approach", in [22], pp. 101–116.

[47] Object Management Group, *OMG Unified Modeling Language Specification*, version 1.3, June 1999, *http://www.omg.org*

[48] P. Pauen, V. Voss and H.-W. Six, "Modelling hypermedia applications with HyDev", eds. A. A. Sutcliffe, J. Ziegler and P. Johnson, *Designing Effective and Usable Multimedia Systems, Proceedings of the IFIP 13.2 Working Conference* (Kluwer, 1998).

[49] J. Rumbaugh, I. Jacobson and G. Booch, *The Unified Modeling Language Reference Manual*, *Object Technology Series* (Addison-Wesley, Reading, MA, 1999).

[50] S. Sauer and G. Engels, "Extending UML for modeling of multimedia applications", eds. M. Hirakawa and P. Mussio, *Proceedings of the IEEE Symposium on Visual Languages* (*VL'99*), (IEEE Computer Society, 1999) 80–87.

[51] D. Schwabe and G. Rossi, "An object oriented approach to web-based applications design", *Theory and Practice of Object Systems* **4**, no. 4 (1998) 207–225.

[52] T. K. Shih, S.-K. Chang, and P. Shih, "A web document development paradigm and its supporting environment", *Proceedings of the 6th International Conference on Distributed Multimedia Systems* (*DMS'99*), 1999.

[53] P. P. da Silva and N. W. Paton, "UMLi: The Unified Modeling Language for interactive applications", in [22], pp. 117–132.

[54] B. C. Smith, L. A. Rowe, J. A. Konstan and K. D. Patel, "The Berkeley Continuous Media Toolkit", *Proceedings of the 4th ACM International Conference on Multimedia'96* (ACM Press, 1996) 451–452.

[55] L. F. G. Soares, R. F. Rodrigues and D. C. Muchaluat Saade, "Modeling, authoring and formatting hypermedia documents in the HyperProp system", *Multimedia Systems* **8**, no. 2 (2000) 118–134.

[56] J. Song, G. Ramalingam, R. E. Miller and B.-K. Yi, "Interactive authoring of multimedia documents in a constraint-based authoring system", *Multimedia Systems* **7**, no. 5 (1999) 424–437.

[57] Sun Microsystems, Java Media API, *http://java.sun.com/products/java-media/*

[58] Sun Microsystems, Java Media Framework, *http://java.sun.com/products/java-media/jmf/*

[59] M. Vazirgiannis, *Interactive Multimedia Documents — Modeling, Authoring, and Implementation Experiences*, *Lecture Notes in Computer Science 1564* (Springer, Berlin, 1999).

[60] P. Vilain, D. Schwabe and C. S. de Souza, "A diagrammatic tool for representing user interaction in UML", in [22], pp. 133–147.

[61] T. Wahl and K. Rothermel, "Representing time in multimedia systems", *Proceedings of the IEEE 1st International Conference on Multimedia Computing and Systems* (*ICMCS'94*) (IEEE Computer Society, 1994) 538–543.

[62] World Wide Web Consortium (W3C), Synchronized Multimedia Integration Language (SMIL), *http://www.w3.org/AudioVideo/*

[63] A. Zamperoni, "GRIDS — Graph-based integrated development of software: Integrating different perspectives of software engineering", *Proceedings of the 18th International Conference on Software Engineering* (*ICSE*) (IEEE Computer Society, 1996) 48–59.