Abstractions in Multimedia Authoring: The MAVA Approach

Jürgen Hauser, Jing Tian

Institute of Parallel and Distributed High-Performance Systems (IPVR)
University of Stuttgart, Breitwiesenstr. 20-22, D - 70565 Stuttgart, Germany
{juergen.hauser, jing.tian}@informatik.uni-stuttgart.de

Abstract. The support of abstractions in authoring systems increases the efficiency of the specification of multimedia documents. Authoring based on a very low abstraction, for example script programming, is very complex and time consuming. In addition, authoring systems supporting a high abstraction are so far restricted to one specific application area. This paper presents an extensible approach that supports abstractions in different application areas (such as computer-based training or a travel guide).
In this approach a meta document model and the corresponding presentation and authoring system are developed. During conception of the authoring system, different concepts (e.g. templates or application specific views) have been realized to introduce further abstractions.

1. Introduction

Although different multimedia document standards [1, 5], research approaches [3, 2, 6] and commercial products [14] have been proposed, the specification of multimedia documents and multimedia authoring are still very complex tasks. An investigation of existing approaches shows that so far multimedia authoring concentrates mainly on the support of the temporal and spatial layout of multimedia documents. Starting with interaction, things tend to become difficult and sometimes the use of scripting languages cannot be avoided [4].

Multimedia documents in different application areas differ in the way users interact with them and how documents react to this interaction. For example, a question in a computer-based training system can be answered either right or wrong. Depending on the answer a different reaction is expected. In case of a right answer, the user gets a compliment, while in case of a wrong answer he or she gets an explanation of the correct answer.

A tourist guide as another example is a multimedia presentation that provides information about particular places of interest. In a tourist guide, the current physical position of the tourist (e.g. in a city or on an island) determines what has to be presented. A change of position is an interaction with a multimedia presentation, so if the user moves into the proximity of a place of interest, the presentation of the corresponding part will be started.

To simplify authoring, an authoring tool has to provide abstractions for particular application areas. These abstractions simplify authoring and thus save time. A second requirement is that it should be possible to use the same authoring tool independently of the application area. And last but not least, it should be possible to add support for new application areas. The last two requirements cannot be satisfied by providing a particular authoring system for each application area (such as Macromedia Authorware).

In this paper an approach that supports authoring in different application areas is presented. Therefore the approach introduces a meta document language that has concrete instances for each application area. The meta language is based on an operator approach where relations between media items are expressed through operators. Application specific operators are a first step in providing a higher abstraction level for multimedia authoring.

A major aspect of an authoring tool is the design of a user interface that is used for authoring documents. It is very important that only a few interaction steps are required during authoring. A specification language on a high abstraction level keeps the authoring process simple in contrast to approaches with a low abstraction level (e.g. script programming). Abstractions and concepts of the authoring tool, for example a template mechanism, further simplify the authoring process.

The MAVA approach relies on the availability of abstractions (in terms of a language or a model) for an application area. If an abstraction is not available, it has to be realized by a MAVA expert, who has programming skills and knowledge about the MAVA presentation system. In practice, however, this is not a big disadvantage of the approach, because reuse of previously developed extensions is expected. Hence, the more extensions have been developed, the less development of new ones is required.

This paper gives an overview of the implementation of the MAVA presentation system and MAVA authoring tool. The realization of the MAVA system is based on Java and the Java Media Framework (JMF).

The results presented in this paper are achieved through the research project MAVA (*M*ultimedi*a* Document *V*ersatile *A*rchitecture), which is supported by a research initiative of the German Research Foundation (DFG) called "Distributed Processing and Exchange of Digital Documents (V3D2)" [13].

The paper is organized as follows. Related work is discussed in section 2. Section 3 introduces the meta document model used in the MAVA approach. In section 4 an authoring tool based on the meta model is presented. Section 5 discusses abstractions based on the model that were integrated into the authoring tool to support authors during specification. In section 6 the architectures of the presentation system and the authoring tool are presented together with details of the implementation. Section 7 summarizes this paper.

2. Related Work

There are two major alternatives for the specification of multimedia documents. The first is to provide a generic language which increases the effort for specification. Another alternative is to provide a specific presentation system for each application area.

A generic system provides a programming or scripting language for the specification of application specific functionality. In such a scripting language there is no support or model for any application area. Examples for presentation systems that provide a scripting language are MHGE5+MEHG6 [5] or Macromedia Director and its scripting language Lingo [14]. An application specific model provides a higher abstraction and therefore reduces the effort needed to specify a presentation. The realization of the semantics is encapsulated in components. Components are reused by the specification language model. Hence, the realization of the semantics and the descriptive specification are separated. Additional concepts, such as a library for scripts, simplify script programming, but they do not provide an application specific model which supports authors during specification.

In contrast, the MAVA approach hides how semantics are realized from authors. Authors just want to specify what the document should look like and not how its semantics are realized, as it is the case if script programming is used.

Another approach is the development of individual presentation systems for each application area. This approach leads to a big overhead for users and content providers because they have to know these systems in detail for content creation and support of users of these systems. Additionally, there is a big effort if each presentation and authoring system has to be developed from scratch. Examples for such an approach are Macromedia Authorware or Question Mark Designer. Also all approaches which provide a fixed language belong to this category (e.g. SMIL 2.0 [1], TIEMPO [3] or Madeus [2]). Their use is limited to the application area defined by their language concepts. For example it is impossible to specify a multiple choice question in SMIL 2.0 that gives the user a certain amount of time so that the user can alter his or her answer within that time. The reason is that the SMIL 2.0 language does not cover such an application area.

In [6] an approach is presented which allows the mapping different temporal concepts on a generic internal representation for temporal concepts. This is a first step towards enlarging the applicability of a system. But not all concepts are time dependent and therefore different models are reasonable. For example, the current position of a user determined by GPS (Global Positioning System) as another way to interact with a presentation leads to a model which depends on the spatial position of the user but not on time. Such a concept can be used in multimedia travel guides as described in the introduction.

3. A Meta Document Model Supporting Abstractions

Instead of using a generic language like a script language, the MAVA approach uses an operator-based language on a high abstraction level. In an operator-based approach, operators are used to specify relations between media items. A document consists of a set of media items and relations between them. A presentation system ensures that relations specified by authors are maintained during presentation. In that sense authoring is the selection of media items and the specification of relations between them.

Apart from the two basic document elements, media items and operators, there is an additional element, a so-called container element. A container is an element that contains operators and media items. Hence, a container is a further abstraction in the specification process. Logical parts of the document specification are grouped together and replaced by a container element. For example, the specification of a question in a computer-based training application can be placed into a container. The next example makes use of a container, namely a *question* container.

For an application area the meta document model has to be instantiated. This means that there are concrete media items, operators and containers for an application area. Fig. 1 shows two sample specifications for the examples mentioned in the introduction section where each example uses a different language.

In Fig. 1 operators are represented by rectangles containing the operator name. Containers are represented by a rectangle that has a shadow. Fig. 1a represents the computer-based training application. In this example a question is related with two scene containers. One represents the congratulation scene and the other one is the explanation scene. The specification has two relations which are represented by a *correct* operator and a *false* operator. The *correct* operator is used to specify which scene has to be presented when the answers is right, the *false* operator specifies which scene has to be presented when the answer is wrong. It is important to distinguish between source and destination media items because the *question* container is presented before the *congrats* or *explanation* scene. Hence, the *question* container is the source media item and the *congrats* scene the destination media. The question is also specified in a descriptive language (not shown in Figure 1). The specification is on a high abstraction level because the author does not need to specify how it is determined whether the answer is right or not.

Fig. 1b shows the example of a tourist guide. A document has always a root container. In case of a tourist guide this is a *LocationArea* container. The container type is denoted in angel brackets '<' and '>'. A *LocationArea* container includes *Location* containers wherein the presentation for particular places of interest is specified. In the example, the *LocationArea* contains one *Location* and a *setLocation* and *circle* operator. With the *setLocation* operator the presentation of the place of interest is situated on a map. The *circle* operator determines the maximum distance between the user and the place of interest to allow the presentation of the *Location* to begin. The exact distance is given by a parameter of the *circle* operator.

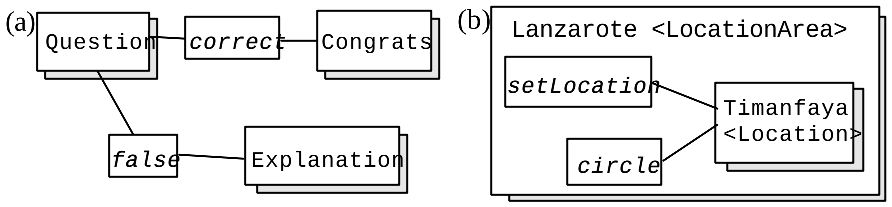

Fig. 1 **Examples for specifications in two different application areas**

The realization of the semantics of the document model is very important. For the realization of semantics so-called managers have been introduced. A manager is responsible for the realization of the semantics of a set of operators. Corresponding media and container components realize the semantics of media items and containers. For details about the realization of the semantics please refer to [10, 11].

4. An Authoring Tool

The *M*AVA *Ed*itor (called MED) is an authoring system for MAVA documents. It visualizes the meta document model presented in the previous section and the necessary interaction steps for authoring MAVA documents. The graphical user interface makes use of the properties of the meta document model that are independent of the application area.

Fig. 2 gives an impression what the user interface of MED looks like. During authoring the author works with different windows. The most important window is the so-called icon view window. An icon view window represents one single container of the document. The window in the center of Fig. 2 (with the title "Timanfaya <Location>") shows the mapping of the abstract document model on icons. The *Location* container comprises two media items and two operators. The *center* operator specifies that the video is centered in the presentation area, whereas the *while* operator [3] specifies that the video and the audio are presented in parallel.

To the left of the document window there are two windows for the selection of media items (top left) and operators (bottom left).

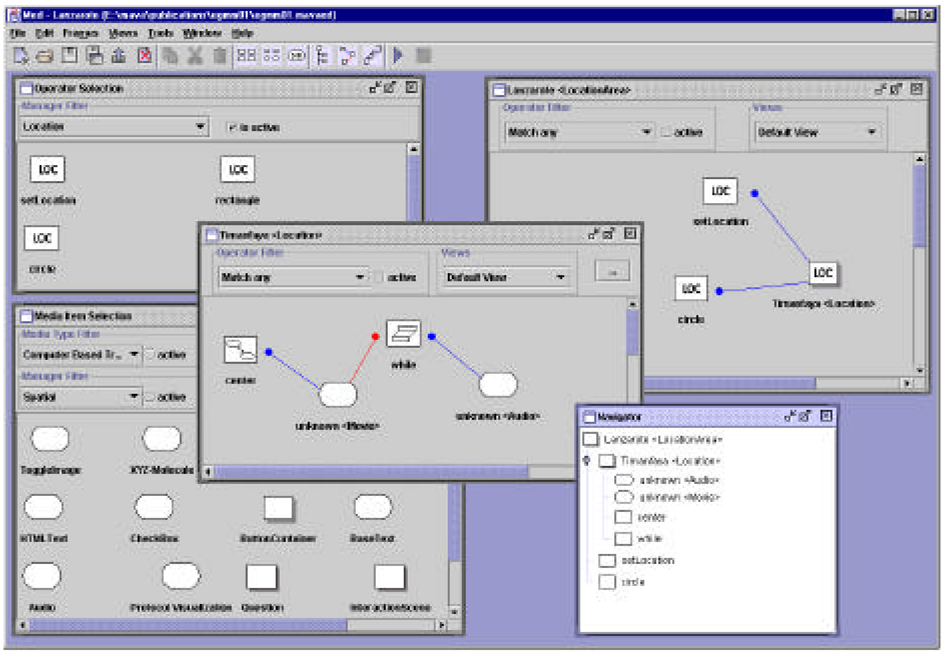

Fig. 2 ***A screenshot from MED***

A document is specified by adding operators and media items and connecting operators to media items. MED provides all available operators, media items and containers in two windows (operator selection window and media item selection window). These windows provide filtering mechanisms to mask out operators which do not belong to a certain concept (e.g. temporal operators).

To add an operator to the document, an operator icon is dragged from the operator selection window and dropped into an icon view window. Media items are added in the same way. Parameters of media items and operators can be specified by so-called properties sheets which are invoked with the right mouse button.

A relation between media items is established by the connection of source and destination media items with an operator. To the left and the right of the operator there is a bullet as a user interface element to enable the connection of operators with media items. The bullet to the left is used for connections with source media, and the bullet to the right is connected to destination media. To connect a media item to an operator the author presses the mouse button on a bullet, moves the mouse pointer to the media icon and then releases the mouse button.

There are also operators which require only destination media items, for example the *center* operator which determines that a media item is spatially centered in the presentation area. Operators without source media just have a bullet to the right for the connection to destination media.

5. Abstractions in the Authoring Tool

Abstraction can be found not only in the meta document model, but also in the authoring system, in order to make authoring of documents easier and more intuitive for authors.

5.1 Navigation

For small multimedia documents, the structure of documents is straightforward for the author. But when the document becomes more complex, there will be many operators and media items in the document, and it will become difficult for the author to keep track of the document's structure. To make the navigation in documents clear, MED provides two concepts: a tree view and a filtering mechanism.

The tree view presents the document in form of a tree as an alternative for the visualization of documents in MED. The root container of the document is the root of the tree, the tree structure is obtained from the containment hierarchy introduced by containers. All containers of the document are nodes of the tree that can be expanded or collapsed. An author can open the icon view of a container or bring the corresponding window to the front by clicking on the corresponding container node in the tree view, allowing easy navigation through the document structure. An example of the tree view window of a document is shown in the bottom right of Fig. 2.

In addition, MED provides a filter mechanism in the icon view of a container. This helps an author to keep track of the specification of one single container. Normally, a container comprises operators from different concepts. The goal of operator filters is to mask out operators in the icon view, which do not belong to a particular concept. For example, if an author processes the temporal aspect of a document, he or she can mask out all operators that do not belong to the temporal concept. Hence, the author can concentrate on the temporal specification.

5.2 Templates

Templates are modules or patterns for regular reuse. The goal of templates in multimedia authoring is to improve reusability in multimedia document design. Compared to the systematic reuse in the field of software engineering [7], many authors still have to build multimedia documents from scratch [9]. To help authors, MED supports document templates as well.

Templates in MED not only improve reusability of authoring work, they also enable authors to design a document at a high level of abstraction. A part of the document, for example a multiple-choice question, can be saved as a template for later reuse. Since multiple-choice questions basically have the same structure, authors reuse a template each time. Hence, they no longer have to build the structure from scratch.

Some authoring systems such as Macromedia Flash, IBM Hotmedia and Tribeworks iShell do not support templates, which makes reuse of multimedia authoring very difficult. Although some authoring systems do support templates, e.g. Macromedia Authorware and ToolBook, there are still some common deficiencies in the template support of these authoring systems. On the one hand, the creation of sophisticated templates in these authoring systems still demands fundamental programming knowledge for script languages like Lingo and OpenScript. Since authors are not expected to be programmers, this requirement for programming knowledge has to be avoided. On the other hand, if the structural information is not visible and not clearly separated from the media infomation of the document, efficient reuse of parts of the document as templates is prevented.

The two problems described above can be avoided in templates provided by MED. Benefiting from the operator approach; the structural information in MAVA documents is visible and clearly separated from the media information of the document. The process of specifying templates in MED is the same as in normal documents. Any part of the document can be saved as a template for later reuse.

5.3 Application Specific Views

Further abstractions to simplify authoring are application specific views. Such a view makes an abstraction from the operators of a language and provides a graphical user interface.

For example, suppose there is the need to describe the spatial layout of a document in absolute coordinates. The required operators are *setPosition* and *setSize*. The specification of the spatial layout consists of several operators and attributes that have to be added step by step, but the spatial layout can be specified in a WYSIWYG view. A graphical visualization of the presentation area enables interactive positioning and resizing of media items. Dragging a media item to a new position determines the attributes for a *setPosition* operator. Hence, such an application specific view will create operators and alter attributes of operators automatically.

It is not possible to provide such views for all concepts. For example for temporal interval operators [5], such a view cannot be provided. It is also difficult to provide such a view for the relative spatial positioning of media items.

When possible, an additional view should be provided for all required specifications. This is not a must because documents can be specified on operator level anyway.

6. Implementation

Architectures for both the MAVA presentation system and the MAVA editor make use of similarities of the specification language in form of the meta document model. The meta document model leads to an implementation of the MAVA approach as a framework that can be extended by components. Components are used to realize the semantics of concrete specification languages (e.g. a manager for a set of operators).

6.1 MAVA Presentation System

Fig. 3 shows the architecture of the MAVA presentation system as it is implemented in the prototype. The core of the presentation system is the MAVA engine, which includes all functionality that is independent of the application area of a document. Examples for such functionality are document loading and class loading. The class manager loads all components that are required for the presentation of a document and caches them for optimization reasons if the extension is used in another document.

The MAVA presentation system provides a framework which is extended by different components loaded by the class manager. These components comprise operators and managers for the realization of a set of operators in a particular application area.

Media items are realized by media objects and media viewers. Media viewers are responsible for the presentation of a media item (e.g. a MPEG renderer) and media objects are responsible for the logical functionality of a media item. A multiple-choice question uses a check box button that has a state. The state describes whether the check box is selected or not. The state and the behavior of the check box are its logical functionality.

The MAVA presentation system is an extensible multimedia presentation system because all required media items, operators and managers for the presentation of a document are dynamically loaded prior to the presentation.

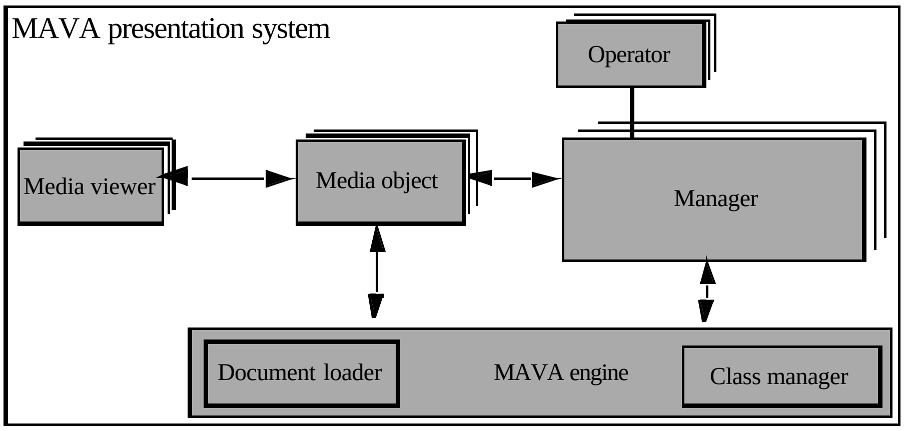

Fig. 3 ***Architecture of the MAVA presentation system***

6.2 MAVA Editor

The core of the MAVA document Editor Architecture is the DEVR (document elements and views repository) and the editor document model (Fig. 4). The editor document model includes all functionality for an internal representation of a document. A document specification is a graph where media items and operators are vertices and connections of operators with media items are edges. For operators the functionality includes handling of references to media items, for containers the management of adding and removing of elements and for all elements the handling of their attributes.

The DEVR holds information on which specification languages can be used currently. Based on the DEVR, the type of actual document elements that are represented as vertices in the document graph are managed. For example the vertex of a media item has a reference to the type description (e.g. an audio medium) in the DEVR. The description from the DEVR describes which attributes are available, like the URL of the media data or the volume during play-out of an audio. The operator description also indicates to which types of media items an operator can be connected.

Based on the editor document model there are three different models that are part of a model-view-controller construct [8]. These models provide different views on the original editor document model. For example the XML model describes the mapping of the document model onto XML. The current implementation uses a graph representation and not a tree-based XML representation for reasons of simplicity (i.e. the document is a graph). Using an XML representation would make the XML model unnecessary but would complicate other modules. Hence, the graph is mapped only for storage in XML. A detailed description of the XML format used for storing and transferring MAVA documents can be found in [12].

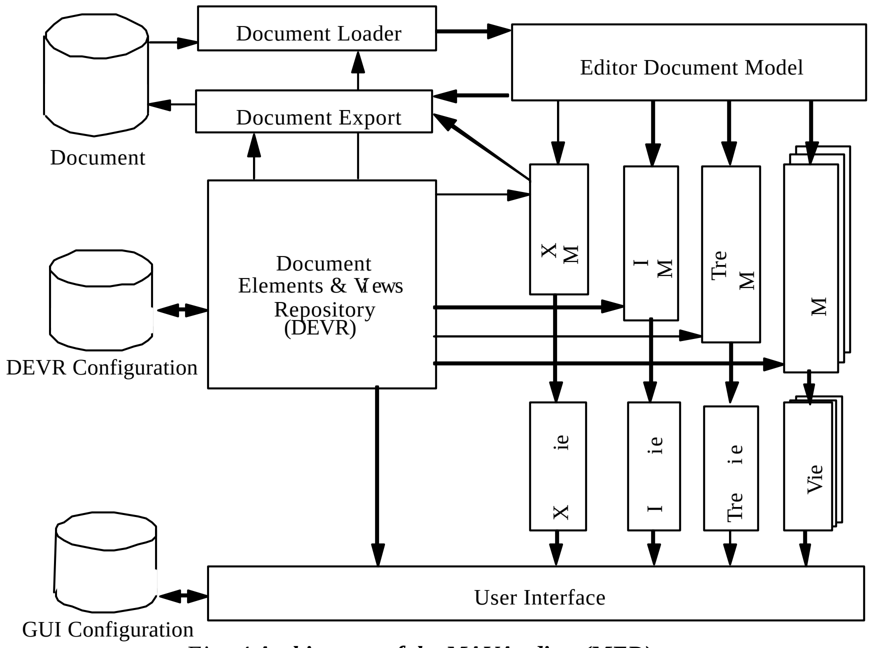

Fig. 4 ***Architecture of the MAVA editor (MED)***

The icon model describes all properties of the document model for the icon view in which an author uses drag and drop for editing a document. The tree model provides the model used for the tree view that supports navigation. For all these models there is also a view component that is responsible for the view and control of the corresponding model. It is also possible to integrate application specific views as described in section 5.3. The availability of application specific views is described by the DEVR. All views are integrated in the graphical user interface.

The last two important modules are the document loader and the document export. The document export module stores the XML model in a file. The document loader realizes the creation of the editor document model based on an XML file that represents a MAVA document.

The current configuration of MED is stored in two configuration files. The DEVR configuration stores the content of the DEVR and the GUI configuration stores the configuration of the graphical user interface, such as window size and working directory.

7. Summary

This paper presents an operator based approach for multimedia authoring. The main goal of the MAVA approach is to support extensibility of the multimedia system with specification languages for different application areas. This is achieved by application specific models (e.g. a computer based training or a travel guide) that provide a high level of abstraction. For authoring, these models are represented by operators. Operators are a mechanism to reuse application specific functionality and templates are used to reuse parts of existing documents in other documents.

An authoring tool that allows interactive document specification is also presented. Throughout the paper different levels of abstraction of the approach are shown and discussed. In practice, high level of abstractions result in reduced effort for authors and therefore reduced costs for multimedia design. Such abstractions are based on reuse of either an application specific language or work done by other authors (e.g. reuse of templates).

References

1. Hoschka, P. et al. (Eds.) "Synchronized Multimedia Integration Language (SMIL) 2.0 Specification". W3C Proposed Recommendation. W3C. 6/2001.
2. Jourdan, M., Layaida N., Roisin, C., Sabry-Ismail, L., Tardif, L. "Madeus, An Authoring Environment for Interactive Multimedia Documents". In Proceedings of the ACM int. conference on Multimedia, Bristol, UK, pp 267-272, 1998.
3. Wirag, S. "Modeling of Adaptable Multimedia Documents". In Proceedings of the 4th International Workshop Interactive Distributed Multimedia Systems and Telecommunication Services, (IDMS'97), Darmstadt, Germany, pp. 220-230,9/1997.
4. ISO/IEC DIS 13522-5."Information Technology - Coding of Multimedia and Hypermedia Information - Part 5". 1995.
5. Allen, J.F. "Maintaining Knowledge about Temporal Intervals". In Communication of the ACM, 26(11):832-843, 11/1983.
6. Jourdan, M., Roisin, C., Tardif, L. "A scalable Toolkit for Designing Multimedia Authoring Environments". In special issue "Multimedia Authoring and Presentation: Strategies, Tools and Experiences" Multimedia Tools and Applications Journal. Kluwer Academic Publishers, 2/3(12):257-279. 1999.
7. Gamma, R. Helm, R. Johnson and J. Vissides: "Design Patterns: Elements of reusable object-oriented software", Addison Wesley, 1995.
8. Goldberg, A. "Smalltalk-80: The Interactive Programming Environment". Addison-Wesley Reading, 1984.
9. Marc Nanard, Jocelyne Nanard and Paul Kahn; "Pushing Reuse in Hypermedia Design: Golden Rules, Design Patterns and Constructive Templates", In Proceedings of the ninth ACM conference on Hypertext and Hypermedia, pp. 11-20, 6/1998.
10. Hauser, J. "Realization of an Extensible Multimedia Document Model", In Proceedings of Eurographics Multimedia '99". Milan, Italy, pp. 113-120, 9/1999.
11. Hauser, J. and Rothermel, K. "Specification and Implementation of an Extensible Multimedia System. In Proceedings of IDMS 2000, LNCS 1905, pp 241-253, Springer, 10/2000.
12. Hauser, J. "A Component-based Extensible Multimedia System" in Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications PDPTA'01, pp. 1243-1249, 6/2001.
13. German Research Foundation (DFG). "Distributed Processing and Exchange of Digital Documents (V3D2)". URL: http://www.cg.cs.tu-bs.de/dfgspp.VVVDD.
14. "Director version 8.5". URL: http://www.macromedia.com/products/director. Macromedia. 2001.