

A MULTIMEDIA TITLE DEVELOPMENT ENVIRONMENT (MTDE)⁺⁺

APHRODITE TSALGATIDOU¹, CONSTANTINE HALATSIS²,
MYRA SPILIOPOULOU³ and MICHAEL HATZOPOULOS⁴

Dept. of Informatics, University of Athens
TYPA Buildings, Panepistimiopolis, Ilisia,
Athens GR-157 71, Greece
email: [1afrodite, 2halatsis, 3myra, 4mike] @ di.uoa.gr

Abstract

The development of Multimedia Titles (M-Titles), i.e. multimedia applications stored on CDs, is performed by many designers and implementors using tools which run on specific platforms and often have complementary and/or overlapping functionality. The potential designer or implementor of a M-Title is restricted by the availability of tools on a selected development platform. This paper presents a Multimedia Title Development Environment (MTDE) which integrates multimedia information, tools used to produce it, as well as their formats and storage media as objects in an Asset Repository. The cooperative construction of complex multimedia objects is modelled as a DAG consisting of two types of nodes: multimedia objects and actions applied to multimedia objects. A M-Title consists of a number of complex multimedia objects linked with each other in a hypernetwork of relationships. MTDE enables a group of authors to cooperatively construct M-Titles as a hypernetwork of multimedia nodes by activating tools running on various platforms, by exploiting the hypermedia functionality of MTDE and by reusing existing material from the Asset Repository.

1. INTRODUCTION

Multimedia technology is used in many areas of human interaction and cooperative work. An appealing example is the development of *Multimedia Titles (M-Titles)*, i.e. multimedia applications, like for example tourist guides, encyclopaedias and multimedia journals, stored on CDs. However, M-Titles are not as widespread as their potential functionality would have suggested due to the fact that they are mostly developed as standalone, non-portable systems,

⁺⁺ Appeared in *Information Processing and Management*, Pergamon Press, vol. 31, no. 1, 1995, pp. 101-112.

incompatible with existing ones. Their integration with other information sources, e.g. data base management systems or information retrieval systems, is accomplished only if specific techniques are used during M-Title development, as the ones reported by deBra & Houben (1992). Furthermore, the incompatibility between the large number of different M-Title development tools which run on different platforms and have complementary and often overlapping functionality, confuses the potential designer or implementor and restricts the M-Title development.

Therefore, a desirable M-Title development environment is one that demonstrates the following features:

- ñ it integrates M-Title development tools and multimedia information in a uniform framework
- ñ it allows the development of M-Titles using appropriate tools running on various platforms
- ñ it controls the invocation of tools and allows them to uniformly access multimedia data
- ñ it provides an attractive user interface which allows an implementor or a designer to browse and inspect the multimedia information
- ñ it allows a team of designers and implementors to re-use existing material and cooperate for the development of a M-Title; designers and implementors may concurrently perform multiple tasks, therefore facilities to guarantee consistency under concurrent operations have to be provided
- ñ it incorporates a versioning mechanism to support the evolutionary development of a M-Title
- ñ it can operate on a heterogeneous client-server architecture
- ñ it handles multimedia data efficiently; this does not depend only on software engineering practices during implementation, but also on certain decisions that have to be taken during the conceptual design of this environment.

To this end, this paper presents a Multimedia Title Development Environment (MTDE). MTDE integrates multimedia objects (i.e., video, image, audio, text, etc.), M-Titles (which are a hyper-network of synchronised multimedia objects), and M-Title development tools running on various platforms, in a uniform framework. Furthermore, MTDE provides facilities to enable the management of the M-Title development process. The support and management of M-Title development is not an easy task, due to the large number of information pieces of different abstraction levels that have to be designed and constructed by a team of designers and implementors working cooperatively. From this point of view, the design of MTDE has the problems and difficulties encountered in the design of a VLSI design environment (Wolf et al, 1988). MTDE can also be compared with CASE tools used for software development. Although there is a large number of CASE tools, there is a lack of tools and environments for supporting and automating the M-Title development process and the work on MTDE attempts to contribute in filling this gap. The architecture of MTDE and the various MTDE components are described in the following section.

2. THE ARCHITECTURE OF MTDE

The architecture of MTDE is depicted in fig. 1. It can be seen that MTDE is used by three types of users: *project managers* of the M-Title development process, M-Title *designers* and M-Title *implementors*. The various MTDE modules, that appear in fig. 1, provide support for all the activities performed by these three different types of users. More specifically, the *M-Title Management* module (*MTM* module) assists *project managers* to distribute M-Title development work to a group of designers and to control their cooperation. The *Hypermedia Functionality & M-Title Development* module (*HMD* module) supports activities usually performed by *project managers* and *designers*, while the *Construction of DAGs & CMM-Objects* module (*CDAG* module) assists *designers* and *implementors* in their various activities.

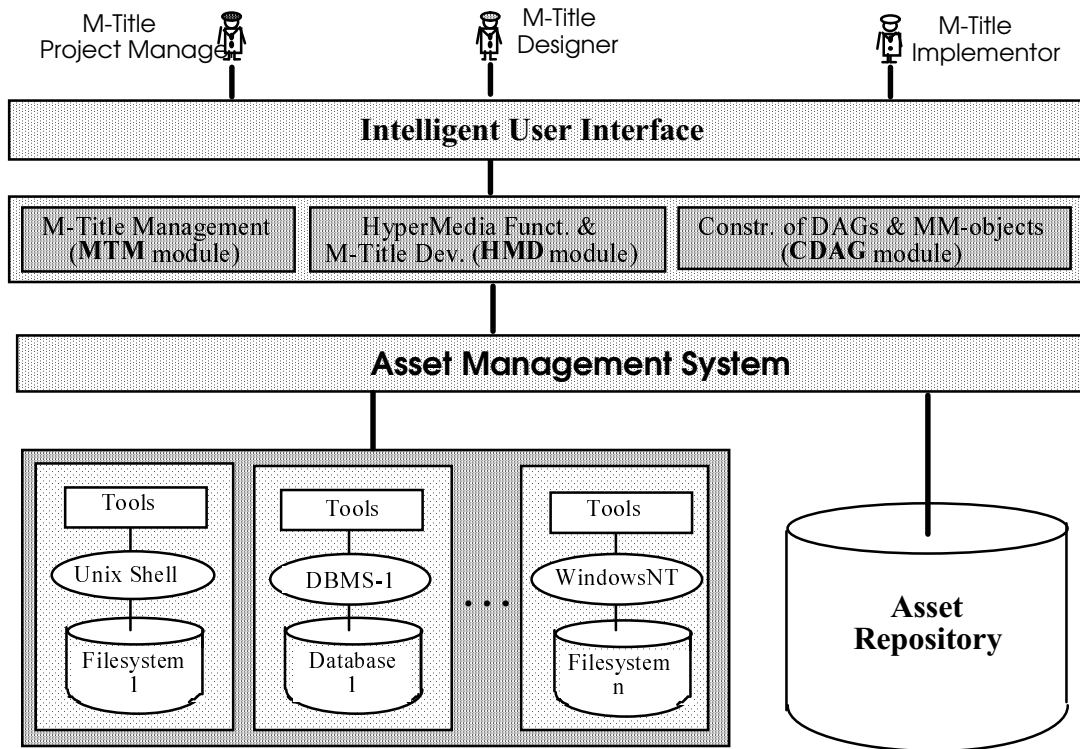


Fig. 1. The architecture of MTDE.

In order to further clarify the support provided by the aforementioned three modules, we will briefly describe a M-Title development process. Thus, a M-Title is viewed as a hypernetwork of multimedia nodes. The design of a M-Title is viewed as a multi-layered hypernetwork, each layer of which corresponds to a different abstraction design level. The nodes of such a hypernetwork are multimedia objects or other hypermedia networks, which, at lower abstraction levels, are decomposed into other hypernetworks. At the lowest abstraction level, each hypernetwork node is a multimedia object.

The first levels of a design hypernetwork for a M-Title are constructed by a *project manager* using the *HMD* module, who subsequently uses the *MTM* module in order to distribute work to a team of *designers* according to the constructed network. Each *designer* may then use the *HMD* module to further decompose the part(s) of the hypernetwork s/he is responsible for, until s/he reaches the lowest abstraction level where all the hypernetwork nodes are multimedia objects (atomic or complex ones) to be constructed.

The construction of complex multimedia objects (CMM-objects) is modelled by a DAG (analysed in following sections), which is constructed by *designers* using the *CDAG* module of MTDE. Thus, at the lowest abstraction design level, the nodes of the hypernetwork are either atomic multimedia objects or DAGs of CMM-objects (see fig. 4). The *CDAG* module is also used by the third type of users, the *implementors*, who construct the various multimedia objects; in case of CMM-objects, implementors construct them according to their DAGs. In order to accomplish this task, implementors may create new or re-use existing material which they synthesize and synchronise using appropriate tools running on different platforms. The cooperation of designers and implementors is controlled by the *MTM* module while the invocation of tools is controlled by the *CDAG* module.

The intelligent user-interface of MTDE assists all types of users during the whole M-Title development process. Thus, it assists: (a) *project managers* in using the *MTM* and *HMD* modules for constructing the initial layers of a M-Title design hypernetwork, for distributing work to designers and generally for managing the M-Title development process, (b) *designers* in using the *HMD* and *CDAG* modules for constructing the rest layers of a M-Title design hypermedia network, and for constructing *DAGs* for CMM-objects, (c) *implementors* in using the *CDAG* module for retrieving existing multimedia objects and/or for invoking tools in order to perform the various interdependent and complementary development activities for the construction of complex multimedia objects according to their DAGs (e.g. select and invoke tools appropriate for the creation of TIFF and YUV images and synthesize them with MIDI audio).

The information pieces integrated in the MTDE framework are called *Assets* and are stored in an *Asset Repository* which is managed by an *Asset Management System*. These assets contain information about the design of a M-Title, the designers involved in it, the components of a M-Title and their construction. In MTDE, all assets are modelled in a hierarchy of object classes. The object-oriented approach was found appropriate for the data model of MTDE for the following reasons: (a) for enabling the uniform modelling of different multimedia information, tools and devices in the form of objects, concealing any specific details inside them; (b) for providing useful abstractions, e.g. specialization, which facilitate the extension of the model and the incorporation of new types of information; (c) for enabling the modelling of the organisation and the interconnection of complex multimedia objects in a high-level language, making the specification easy to comprehend by non-programmers; (d) for providing executable

specifications which can be used for prototyping and validation purposes and for implementation of the final system in an object-oriented language.

The following sections of the paper focus mainly on the aspects of the MTDE that are related to the actual development of M-Titles. Thus, section 3 describes the part of the data model of MTDE which is concerned with meta-information held about the various multimedia data stored in different platforms. Sections 4 and 5 discuss the cooperative construction and updates on the various multimedia objects. The part of the MTDE data model which supports the management of a M-Title development process and is mainly provided by the *MTM* module, as well as information about the intelligent user interface and the other MTDE components are described elsewhere (Tsalgatidou & Halatsis, 1994). Finally, section 6 presents related work and section 7 concludes this work.

3. THE DATA MODEL OF MTDE

The data model of MTDE is represented by a hierarchy of object classes and a property domain graph. The hierarchy of object classes and their properties model all *Assets*, i.e. M-Titles, Atomic and Complex Multimedia Objects, their Formats and storage Media, the Tools used for their development, as well as information about the management of M-Title development projects and the designers involved in them. As it was mentioned in the previous section, classes and properties relevant to the management of the M-Title development process are not shown here. This section deals with classes and properties pertinent to modelling and construction of multimedia documents per se. Fig. 2 depicts part of the hierarchy of these object classes, while figure 3, shows part of the property domain graph which links the properties of the various object classes to their domain classes.

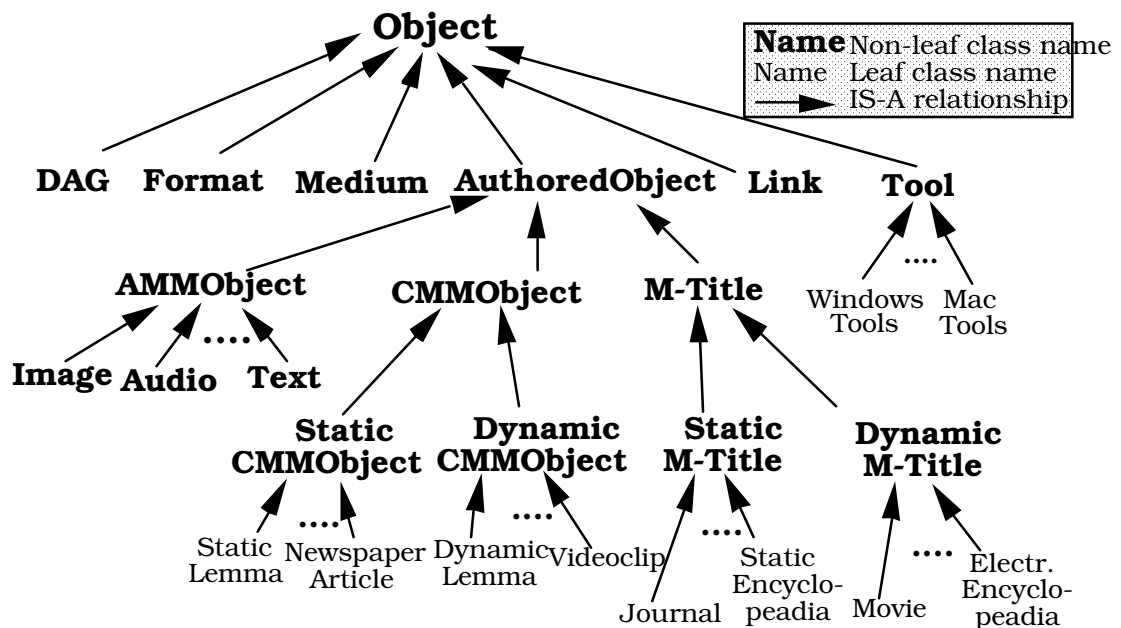


Fig. 2. The hierarchy of object classes

3.1 Atomic Multimedia Objects (AMM-Objects)

AMM-Objects are objects of a simple data type, i.e., text, image, audio, etc., which cannot be further decomposed. Thus, an audio object that cannot be analysed into simpler components is an AMM-Object belonging to the class 'Audio'. AMM-Objects are modelled in a class subhierarchy rooted at the generic class 'AMMObject'. This subhierarchy is decomposed in leaf classes accommodating objects in a variety of formats, produced by specific tools. Such a specialization of classes highlights the ways and restrictions placed on the usage of AMM-Objects, e.g., an instance of the 'MIDI_Audio' class, is handled in a different way than an instance of class 'CD_Audio'. This assists an implementor in identifying an AMM-Object that satisfies his/her requirements (which are expressed in terms of structure, format, etc.) among a variety of similar AMM-Objects.

AMM-Objects are produced by tools which create them or which generate them by decomposing complex multimedia objects into their components. The created AMM-Objects are stored in the environment where the used tool operates. The 'AMMObject' class has a property called "Content" which contains the reference to the AMM-Object storage location and is thus used to incorporate the created AMM-Objects in the MTDE. Another interesting property of AMM-Objects is the "is-component-of" which contains the complex objects in the construction of which an AMM-Object participates and is used for notifying them of any changes occurred in the AMM-Object. The "change-notification-timestamp" is another property which stores the time the object was created or last updated.

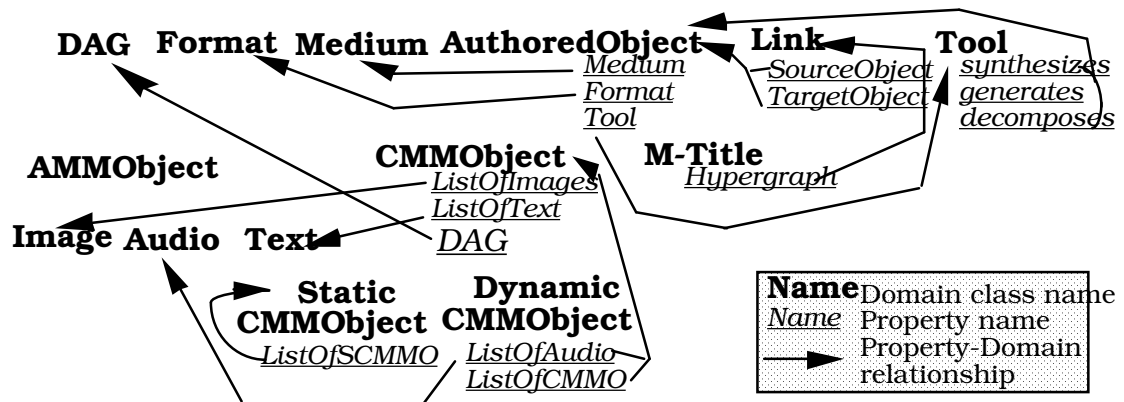


Fig. 3. The Property-Domain Graph

3.2 Complex Multimedia Objects (CMM-Objects) and their DAGs

CMM-Objects are recursively defined as aggregates of CMM- and AMM-Objects. They are maintained in a class subhierarchy rooted at the generic class 'CMMObject'. This class has two subclasses: 'StaticCMMObject' and 'DynamicCMMObject'. The instances of the first subclass consist of text and images arranged in various spatial orientations. Instances of the latter subclass consist of video images, animations etc., incorporating temporal relationships, e.g. relative timing during presentation.

The various actions needed for the construction of a CMM-object, such as: the gathering or capturing of multimedia information, the grouping and organisation of objects into higher order composites, the synchronisation of temporal objects etc., are performed with the use of appropriate tools. These actions are represented in a Direct Acyclic Graph (DAG) which is stored in the "DAG" property of the 'CMMObject' class. A DAG models *the whole construction process* of a CMM-Object (e.g. of an article, a lemma or a videoclip). More about the DAG and the construction of a CMM-Object by multiple implementors is described in the following sections.

Other properties of a CMM-Object hold information about: (a) the "*synthesis*" of its components (i.e., overlay or synchronisation information, etc.), so that it can be always reconstructed, (b) the "*medium*" a CMM-Object is stored, e.g. CD, optical disc, etc. (c) its "*components*", i.e., the AMM- and CMM-Objects used to synthesize it, (d) the CMM-Objects in the construction of which it participates (i.e. the "*is-component-of*" relationship), (e) a "*change-notification-timestamp*" which stores the time the object was created or last updated and (d) the "*change-approval-timestamp*" which holds the time that the CMM-Object accepted the changes occurred in its components. A CMM-Object is *reference-consistent* if none of the objects referenced by it has a change-notification-timestamp which exceeds the change-approval-timestamp of this CMM-Object.

3.3 Multimedia Titles (M-Titles)

M-Titles are designer-defined collections of *static* and/or *dynamic* CMM-Objects. M-Titles accommodate journals, encyclopaedias, books, newspapers etc. A M-Title is an aggregate of CMM-Objects linked according to relationships modelled in class 'Link'. Thus, a M-Title is not a CMM-Object but a network of AMM- and CMM-Objects connected via links (see fig. 4) and is modelled in class 'M-Title'.

The construction of the hypernetwork of a M-Title includes the construction of the nodes of the hypernetwork, and their subsequent interconnection via links. The creation of links is controlled by the *HMD* module of MTDE. An AMM-Object node is created by an implementor with the use of an appropriate tool. A CMM-Object node is created by one or more implementors according to its construction DAG, using one or more tools. Thus, many implementors may collaborate for the construction of individual nodes and the subsequent creation of links between them, in order to create a hypernetwork. This collaboration is mainly controlled by the *MTM* module.

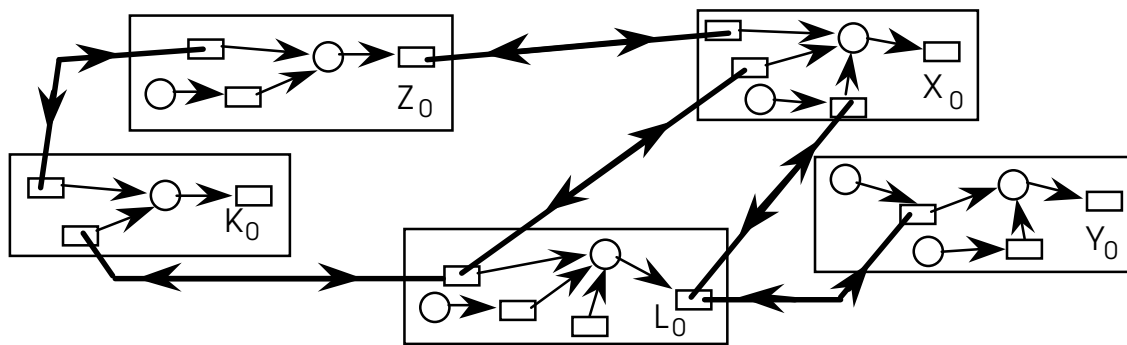


Fig. 4. A hypernetwork of a M-Title (named MT). Nodes are CMM-Objects, the DAGs of which are shown here. Bold lines depict bi-directional links connecting the components of the various nodes.

The constructed M-Title hypernetwork can then be navigated by an end-user. The constituents of a M-Title hold information about their links and about the M-Title in the construction of which they participate. On the other hand, a M-Title holds information about the various nodes constituting it, in the "*Hypergraph*" property provided by class 'M-Title'.

The type of a link connecting two objects may be *structural*, e.g. below-on-page, next-page for articles of a newspaper, or *semantic*, like is-referenced-by between scientific articles and special-case between encyclopaedia lemmas. The 'Link' class has properties for specifying the source and the target object of a link, its anchors (i.e., the exact points inside the source and target objects, where the link emanates from and points to, respectively), and its type and name. Further application-specific properties describing the relationship semantics may be defined by an implementor.

3.4 Tools and Formats

Tools are modelled as abstract facilities, organised into classes of the 'Tool' subhierarchy. The tool classes are specialised according to the platform they run on. This specialization is of high significance to designers and implementors of multiple-platform M-Titles, who need to use platform-specific tools. The 'Tool' class has a number of properties pertinent to the modelled tools. Thus, for each tool, MTDE holds information about the tool version and the types and formats of AMM- and CMM-Objects which can be *created*, *synthesized*, *generated*, *decomposed*, *exported* or *imported* by the tool (e.g., *import* TIFF images and ASCII text and *generate* LATEX document), see for example fig. 5. These properties are useful for the selection of tools that can cooperate for the accomplishment of complementary tasks for the creation of a CMM-Object, e.g. capture an image of a certain format, embed it into a piece of ascii text, synthesize it with a video captured by a video camera and produce a dynamic CMM-Object. The domains of the tool properties are the classes of the 'AuthoredObject' and 'Format' subhierarchies. The classes of 'Format' subhierarchy are organised by specific data types and accommodate object instances describing proprietary and standard formats. Classes of 'Format' subhierarchy serve also as domains for the "*Format*" property of the classes in the 'AuthoredObject' subhierarchy.

```
class Windows_Tools
```

```

instance      tool-w1
name          tool-w1
version       2.0
creates       [[text, ascii, latex, ..], [DynamicCMMObject, format-c1,
..],...]
decomposes    [[image, TIFF,...], [audio, MIDI,...],...]
processes     [text, format-t1, format-t2,...]
imports       [[text, format-t1, ...], [image, format-i1,...], [sound,
format-s1,...],...]
synthesizes   [[image, format-i1, format-i2, ..], [audio, format-s1,
..],...]
generates/exports [[StaticCMMObject, type1,..], [text, format-t1, ..]
...

```

Fig. 5. An instance of the Class 'Windows_Tools'

The selection of appropriate tools and their activation is controlled by the *CDAG* module. When an *implementor* activates a tool, s/he is immediately "transferred" to the environment of the tool. All multimedia data created by a tool are stored locally into the tool environment. Additionally, the implementor creates meta-information about this multimedia data (e.g. the name of the file where it is stored, its links with other objects etc.) using the appropriate MTDE tools and stores it into the Asset Repository.

3.5 Processes, Process Elements and DAGs

In general, a process is a set of partially ordered steps which are followed in order to reach a goal and is composed of *process elements*. A *process model* is an abstract description of an actual or proposed process and represents selected process elements that can be enacted by a human or a machine. *Process script* is a process model to be performed by a human, while *process program* is a process model to be enacted by a machine (Curtis et al. 1992).

In MTDE, the task of creating a CMM-Object is a *construction process* and is modelled by a DAG. A DAG models the actions that have to be performed for the construction of a CMM-Object, as well as the multimedia data used by or produced by these actions. A DAG is not a program to be executed for constructing a CMM-Object; it rather describes a set of steps that an implementor should follow in order to construct a complex object. The implementor will have to retrieve specified objects from the Asset Repository and to activate selected tools in order to perform the steps described by a DAG. Therefore, a DAG is a *process script* rather than a process program.

The design of a CMM-Object construction (i.e. the construction of its DAG) and the actual construction of it, are not usually performed by the same person. A DAG is usually designed by a *designer* using the *CDAG* module, and then a number of *implementors* are responsible for cooperatively constructing the CMM-Object (i.e. for creating its actual content and its links) according to its designed DAG. Cooperation of all these people is controlled by the *MTM* module.

A DAG has two kinds of nodes: *rectangles* which represent AMM- and CMM-Objects, and *circles* which represent *process elements* that specify the sequence of actions, like capture,

synthesize, synchronize etc. These actions have to be performed in order to generate specified multimedia objects. Multiple edges emanating from a multimedia object imply that this object is input to process elements. Multiple edges emanating from (or pointing to) a process element (i.e., circle node) imply that this process element describes actions which produce many multimedia objects (or requires many objects as input to the specified actions, respectively). A leaf circle node, i.e. a circle node without input nodes is a *create process element* which specifies that some data must be captured from the external environment of MTDE, e.g. capture a video object from a video camera, an image from a scanner, etc. A leaf rectangle node is a multimedia object existing in the Asset Repository and is retrieved by an implementor in order to be used as input in subsequent actions. Fig. 6 depicts an example of a DAG.

Actions specified in process elements are described in natural-like language, e.g. *synthesize* YUV images with audio and LATEX text. An implementor starts constructing a CMM-Object by retrieving the leaf object nodes of a DAG (see for example X_3 in fig. 6) and by first executing the actions specified in the various leaf process elements of the DAG (e.g. P_1 , P_3 in fig. 6). The descriptions of process elements in natural-like language, help the implementor to select tools appropriate for the execution of the specified actions. Thus, a DAG guides implementors to construct a CMM-Object, step by step.

4. COOPERATIVE CONSTRUCTION OF CMM-OBJECTS AND M-TITLES

The cooperative construction of CMM-Objects and M-Titles will be discussed using an example of a CMM-Object X_0 , the DAG of which is shown in figure 6. X_0 contains a painting, together with textual information about it and its painter, as well as a picture of the painter and some accompanying music. X_0 is composed of an atomic audio object X_6 , containing the accompanying music, and a static multimedia composite object X_5 . X_5 is composed of two textual atomic objects, X_3 and X_4 - containing information about the painter and the painting, respectively - and of two atomic image objects, X_1 containing an image of the painting, and X_2 containing an image of the painter. The indices of the processes are named after the indices of the objects they generate; thus, object X_1 is generated by process P_1 , object X_5 is generated by process P_5 , and so on.

The CMM-Objects included in the DAG (i.e. X_0 and X_5) contain information about their components, i.e. which they are and how they have been or they will be synthesized. Process elements hold information about which actions have to be executed on which objects and about the type of objects they generate. Multiple implementors may collaborate for the construction of X_0 . Their cooperation is mainly controlled by the *MTM* module of MTDE (Tsalgaidou & Halatsis, 1994). Each implementor is responsible for constructing some of X_0 's constituents, i.e., for executing some of the actions specified in the process elements of its construction DAG. When all the specified actions have been performed, the *change-approval-timestamp* of all generated objects is set to current time and all objects using the generated ones, are notified. An

implementor may store the intermediate multimedia objects, created during X_0 construction, in the Asset Repository for future use.

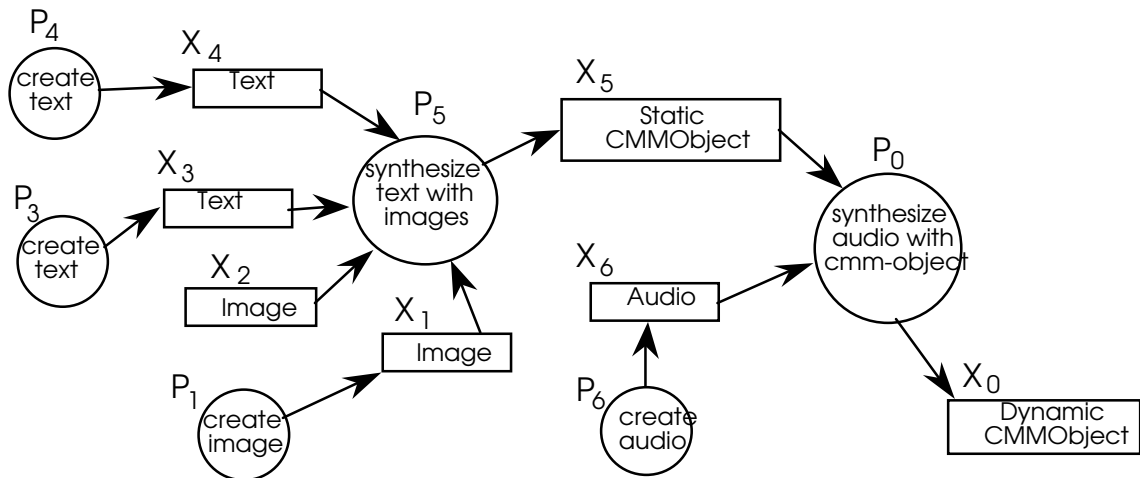


Fig. 6. Example of a DAG for a dynamic complex multimedia object.

Now we suppose that the painting captured in image X_1 , is in a museum of arts. We want to develop a M-Title containing all paintings of the various rooms of the museum. This means that, a hypernetwork, similar to the one in fig. 5, should be constructed, the nodes of which will be CMM-Objects, similar to X_0 . These nodes will be linked with relationships like, painting-in-next-room, painting-in-previous-room, painting-with-the-same-subject etc. An end-user should be able to navigate this network going from one painting to another and obtain any information s/he wishes about the museum and its paintings.

For the construction of this M-Title, a project manager, using the *HMD* module, will design the first abstraction levels of a hypernetwork and then, using the *MTM* module, will distribute work to a group of designers. The designers, using the *HMD* module, will then refine the part of the hypernetwork they are responsible for, until the lowest abstraction level where each node is a multimedia object (either atomic or complex). In case of complex objects, designers should also construct their DAGs, according to which, implementors will then construct the nodes using the *CDAG* module and any appropriate tools from various platforms.

5. UPDATES ON CMM-OBJECTS AND M-TITLES

Updates on the multimedia assets of the Asset Repository are controlled by MTDE and are performed according to a version timestamping technique. Information stored in the *change-notification-timestamp* and in the *change-approval-timestamp* of objects is used for maintaining consistency of the stored assets. The updated objects may be stored as new versions. The versioning mechanism used in MTDE is similar to the one proposed by Oesterbye (1992). Versions are a kind of complex objects, the components of which are all the versions of a certain object containing information about the implementors and time of the change.

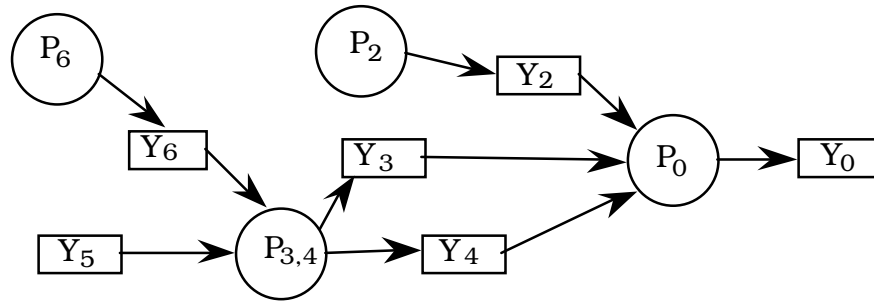


Fig. 7. The DAG of the CMM-Object Y_0 .

Updates on AMM-Objects are performed with the use of selected tools. The CMM-Objects or M-Titles using the updated AMM-Objects are notified and they either accept the updates, according to a procedure described below or they do not accept them and retain the previous version of the updated objects.

Updates on a CMM-Object are either direct updates on it or updates on its components. Direct updates on a CMM-Object, e.g. on Y_0 , see fig. 7, result in a new version of it, which cannot be re-composed out of its components and therefore its DAG is eliminated. The previous version of Y_0 preserves its construction DAG and is held in the Asset Repository. Any objects affected by this update (i.e., CMM-Objects or M-Titles using Y_0 or referring to it, respectively) are notified, so that they decide whether to accept the changes or not.

If the changes take place on the components of Y_0 , for example on Y_5 or on Y_3 , all objects using the updated ones are notified. Thus:

- ñ if the updated component is a leaf, as it is Y_5 in the DAG of Y_0 , then:
 - If the implementor wants to have these changes accepted by the final object Y_0 , these must be first accepted by all intermediate objects. Therefore, the actions specified in $P_{3,4}$ and use Y_5 as input have to be re-executed in order to incorporate Y_5 's updated version. This re-execution will generate new versions of Y_3 and Y_4 and will set their *change-notification-timestamps* to current time, which, in this way, will exceed the *change-approval-timestamp* of Y_0 making it *reference-inconsistent*. Therefore, the actions specified in P_0 will have to be also re-executed in order to reconstruct Y_0 , incorporating Y_3 's and Y_4 's updated versions. Then both the *change-approval-timestamp* and the *change-notification-timestamp* of Y_0 are set to current time, making it *reference-consistent* again.
 - If the implementor of Y_0 does not wish to accept the changes in Y_5 , s/he retains the previous version of Y_5 in Y_0 's DAG and updates the timestamps of the affected objects accordingly, so that all of them are *reference-consistent*.
- ñ if the updated component is not a leaf, as it is Y_3 in Y_0 's DAG, and the implementor wishes to accept these changes, the part of Y_0 's DAG pointing to Y_3 is eliminated, as Y_3 cannot be re-constructed from its components. Thus, Y_3 is substituted in Y_0 's DAG by its new version and

becomes a leaf node. In the sequel, the aforementioned procedure for updates on leaf nodes of a CMM-Object is applied until the new version of Y_0 is constructed.

- ñ if the modified final object Y_0 is a node in the hypernetwork of a M-Title (for example of MT, see fig. 4), then the old version of Y_0 is substituted by its updated version and the affected links are modified accordingly using the facilities provided by MTDE. Both the *change-notification-timestamp* and the *change-approval-timestamp* of MT are set to the current time. However, if the implementor of MT does not want to accept the changes of Y_0 , MT retains the previous version of Y_0 and the notification timestamps of the affected objects are modified accordingly.

It is obvious that the evolutionary construction of a CMM-Object or a M-Title as well as any updates on them by a group of designers or implementors may be a long process. MTDE supervises these processes using the various constructs of the data model and exploiting the timestamping, the versioning and notification mechanisms of the OODBMS on top of which MTDE is built.

6. RELATED WORK

Researchers have lately realised that multimedia application development must be carried out within frameworks and environments that support the modelling of different kinds of multimedia data, as well as their composition and their reuse. An example of such an approach is the one presented by Gibbs (1992) where multimedia applications are interactively constructed by connecting groups of generic components with the use of a tool which is described in (deMey & Gibbs, 1993); the generic multimedia components and their connections are software abstractions provided by an object-oriented framework. Other approaches to multimedia data modelling may be found in (Bertino et al., 1988; Gibbs, 1991; Klas et al., 1990; Masunaga, 1987).

Additionally, extensions to object-oriented databases for providing multimedia support have been earlier defined, e.g. (Woelk & Kim, 1987). Multi-user OODBMSs (e.g. Kim et al, 1989; Bretl et al, 1989) offer an object-oriented framework for multimedia information modelling and provide multi-user support for transaction management and concurrency control. MTDE is therefore being developed on top of a multi-user OODBMS, in order to to exploit the transaction management facilities of the underlying data base system for recovery, as well as for version and concurrency control and, because the information modelled in the object-oriented MTDE data model can be more accurately mapped onto object-oriented implementation structures.

Multimedia title development is conceptually close to the field of hyperdocument development. In both areas, an important issue is the support of versioning, i.e. the support for maintaining the history of changes of the stored objects (Halasz, 1988). Examples of hypertext systems that provide versioning support are: HAM (Cambell & Goodman, 1988) - a general-

purpose transaction-based server for a hypertext storage system which provides an automatic version history mechanism, Neptune (Delisle & Schwartz, 1986) - a hypertext system built using HAM, Intermedia (Yankelovich et al., 1988) and KMS (Acksyn et al., 1988).

Collaborative work is also a major issue in multimedia and hypermedia document development. Rada (1991, pp. 112-142) discusses about the principles of collaborative hypertext and describes some of the systems providing such support. Rodden et al (1992) examine a number of issues (like schema definitions, roles, access control, transaction mechanisms, monitoring and activity support etc.) surrounding the support of cooperative applications. These issues have been taken into account for the design of MTDE. Examples of hypertext or hypermedia systems providing support for collaborative work are all the aforementioned systems (i.e. HAM, Neptune, Intermedia, KMS) and MUCH (Rada & Barlow, 1991).

MTDE is also related to the ThyDoc (Sobiesiak & Mylopoulos, 1991) and SEPIA (Streitz et al., 1992) hypermedia authoring environments. ThyDoc is an *authoring-in-the-large* environment, which treats authoring as a knowledge-acquisition process that captures formally (in terms of a conceptual schema) and informally (through text, graphics, etc.) all the knowledge needed by teams of authors for the analysis, design, development and maintenance of complex hypertext documents. SEPIA is built on top of a multi-user DBMS and provides persistent and shared data storage, hypermedia data model with composites, and authoring functionality and support for cooperative work. A hypermedia server has been developed (Haake, 1992; Haake & Wilson, 1992) to support versioning in the SEPIA environment. A user in SEPIA may cooperate with others either asynchronously (only one author can work on a multi-author document at a given time) or synchronously (two or more authors working on a document at the same time).

MTDE exhibits many of the features of the aforementioned systems, as it was discussed in previous sections. Our work differs from others in the sense that, although it deals with the modelling of the cooperative construction of M-Titles, it also places emphasis on the integration in a unified framework of (a) multimedia information stored in different platforms and b) of the tools producing this information.

7. CONCLUSION

This paper presented a Multimedia Title Development Environment (MTDE), which enables the development of multimedia applications within a uniform framework. This framework integrates multimedia development tools running on different platforms, and multimedia information of various types and formats, in a multimedia *Asset Repository*. Some of the main features and advantages of this approach are the following:

- ñ Developers can cooperate for the evolutionary construction of a M-Title (i.e. of a hypernetwork of CMM-Objects) by re-using material existing in the Asset Repository and/or by creating new multimedia information.
- ñ Multimedia data processing and synchronisation is performed independently by tools running on different platforms designed exactly for this purpose.
- ñ The provided intelligent user interface helps a M-Title developer to select and invoke suitable tools and to assign hypermedia functionality to multimedia assets. It also enables him/her to define additional properties to existing MTDE object classes and to create additional application-specific subclasses in order to refine the assets and their relationship semantics.

The current status of the system is at the end of the design phase and the beginning of the implementation phase. More specifically, implementation of the *HMD* has almost finished and is currently under testing, while implementation of the other modules of MTDE will be starting soon. Future work will focus on further enhancement of the intelligent user interface and will consider query processing and optimisation mechanisms, like the ones reported in (Bretl et al, 1989; Kim, 1989; Stein & Maier, 1991), to ensure satisfactory response for queries which need to resolve partially specified paths within the class hierarchy.

REFERENCES

- Acksyn, R.M., McCracken, D.L. & Yoder, E.A. (1988). KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations. *Communications of the ACM*, 31 (7), 820-835.
- Bertino, E., Rabbiti, F. & Gibbs, S. (1988). Query Processing in a Multimedia Document System. *ACM Transactions On Information Systems*, 6 (1), 1-41.
- Bretl, R., et al. (1989). The GemStone Data Management System. In W. Kim and F.H. Lochovsky (Eds.), *Object-Oriented Concepts, Databases and Applications* (pp. 283-308). ACM press, Addison-Wesley Publishing Co.
- Cambel, B. & Goodman, J.M. (1988). HAM: A General Purpose Hypertext Abstract Machine. *Communications of the ACM*, 31 (7), 856-861.
- Curtis, B., Kellner, M.I. & Over, J. (1992). Process Modelling. *Communications of the ACM*, 35 (9), 75-90.
- deBra, P. & Houben, J.-G. (1992, Dec.). *An Extensible Data Model for Hyperdocuments*. Paper presented at the 4th ACM Conference on Hypertext (ECHT '92), Milano, Italy.
- Delisle, N & Schwartz (1986, May). *Neptune: A Hypertext System for CAD Applications*. Paper presented at the International ACM SIGMOD Conference on Management of Data, Washington D.C.
- deMey, V. & Gibbs, S. (1993, Aug.). *A Multimedia Component Kit*. Paper presented at the ACM Conference on Multimedia 93, Anaheim, CA, USA.
- Gibbs, S. (1991). Composite Multimedia and Active Objects. *ACM SIGPLAN Notices*, 26 (11), 97-112.
- Gibbs, S. (1992). *Application Construction and Component Design in an Object-Oriented Multimedia Framework*. Paper presented at the 3rd Int. Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, USA.

- Haake, A. (1992, Dec.). *CoVer: A Contextual Version Server for Hypertext Applications*. Paper presented at the 4th ACM Conference on Hypertext (ECHT '92), Milano, Italy.
- Haake, J. M. & Wilson, B. (1992, Nov.). *Supporting Collaborative Writing of Hyperdocuments in SEPIA*. Paper presented at the ACM Conference on CSCW, Ontario, Canada.
- Halasz, F.G. (1988). Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM*, 31 (7), 836-852.
- Kifer, W., Kim W., & Sagiv, Y. (1992, June). *Querying Object-Oriented Databases*. Paper presented at the ACM-SIGMOD Conference, California, CA.
- Kim, W., et al (1989). Features of the ORION Object-Oriented Database System. In W. Kim and F.H. Lochovsky (Eds.), *Object-Oriented Databases and Applications* (pp. 251-282). ACM press, Addison-Wesley Publ. Co.
- Klas, W., Neuhold, E.J. & Schrefl, M. (1990). Using an Object-Oriented Approach to Model Multimedia Data. *Computer Communications*, 13 (4), 204-216.
- Masunaga, Y. (1987, April). *Multimedia Databases: A Formal Framework*. Paper presented at the IEEE Computer Society Office Automation Symposium, Gaithersburg, MD, USA.
- Oesterbye, K. (1992, Dec.). *Structural and Cognitive Problems in Providing Version Control for Hypertext*. Paper presented at the 4th ACM Conference on Hypertext (ECHT '92), Milano, Italy.
- Rada, R. (1991). *HYPERTEXT: from Text to Expertext*. McGraw-Hill Book Company, UK.
- Rada, R. & Barlow, J. (1991). Document creation in offices. In P. Saleniaks (Ed.), *Computing Technologies: New Directions and Applications* (pp. 43-74). Ellis Horwood, London.
- Rodden, T, Mariani, M.A. & Blair, G. (1992). Supporting Cooperative Applications. *Computer Supported Cooperative Work (CSCW)*, 1 (1, 2), 41-68.
- Sobiesiak, R. & Mylopoulos, J. (1991). *A Conceptual Modelling Approach to Authoring-in-the-Large for Hypertext Documents*. *SIGOIS Bulletin*, 12 (2, 3), 225-239.
- Stein, J. & Maier, D. (1991). Associative Access Support in GemStone. In Dittrich, K.R., Dayal, U. & Buchman, A.P. (Eds.), *Object-Oriented Database Systems* (pp. 323-339). Springer-Verlag.
- Streitz, N. et al. (1992, Dec.). *SEPIA: A Cooperative Hypermedia Authoring Environment*. Paper presented at the 4th ACM Conference on Hypertext (ECHT '92), Milano, Italy.
- Tsalgatidou, A. & Halatsis, C. (1994). *MTDE: An environment for developing M-Titles and managing their development process*. Technical report, University of Athens, Dept. of Informatics, Athens.
- Woelk, D. & Kim, W. (1987). *Multimedia Information Management in an Object-Oriented Database System*. Paper presented at the 13th VLDB Conference, Brighton, UK.
- Woelf, P. van der, Meijs, N. van der, Leuken, T.G.R. van, Widya, I. & Dewilde, P. (1988). Principles of Open VLSI Data Management in the NELSI Design System. In Hermann, O.E. & Beijnum, B.J.F. van (Eds.), *Lecture Notes of the NELSI Introduction Course* (pp. 12-35), Participants Edition, TU Twente, The Netherlands.
- Yankelovich, N., Haan, B.J., Meyrowitz, N.K. & Drucker, S. (1988). Intermedia: The Concept and the Construction of a Seamless Information Environment. *IEEE Computer*, 21(1), 81-96.