EMMOS: TRADEABLE UNITS OF KNOWLEDGE-ENRICHED MULTIMEDIA CONTENT

Utz Westermann[1], Sonja Zillner[1], Karin Schellner[2], Wolfgang Klas[1,2]

[1] University of Vienna, Dept. of Computer Science and Business Informatics,

Liebiggasse 4, A-1010 Wien, Austria

phone: +43-1-4277-38430 fax: +43-1-4277-38449

{gerd-utz.westermann,sonja.zillner,wolfgang.klas}@univie.ac.at

[2] Research Studios Austria - Digital Memory Engineering

ARC Seibersdorf Research GmbH, Thurngasse 8/20, A-1090 Wien, Austria

phone: +43-1-585-0537 fax: +43-1-585-3741

karin.schellner@researchstudio.at

EMMOS: TRADEABLE UNITS OF KNOWLEDGE-ENRICHED MULTIMEDIA CONTENT

ABSTRACT

Current semantic approaches to multimedia content modeling treat the content's media, the semantic description of the content, and the functionality performed on the content such as rendering as separate entities, usually kept on separate servers in separate files or databases and typically under the control of different authorities. This separation of content from its description and functionality hinders the exchange and sharing of content in collaborative multimedia application scenarios. In this chapter, we propose Enhanced Multimedia Meta Objects (Emmos) as a new content modeling formalism that combines multimedia content with its description and functionality. Emmos can be serialized and exchanged in their entirety - covering media, description, and functionality - and are versionable, thereby establishing a suitable foundation for collaborative multimedia applications. We outline a distributed infrastructure for Emmo management and illustrate the benefits and usefulness of Emmos and this infrastructure by means of two practical applications.

KEYWORDS

Multimedia, Data Semantics, Metadata, Multimedia Databases, Semantic Data Model

INTRODUCTION

Today's multimedia content formats such as HTML (Raggett et al., 1999), SMIL (Ayars et al., 2001), or SVG (Ferraiolo et al., 2003) primarily encode the presentation of content but not the information content conveys. But this *presentation-oriented* modeling only permits the hard-wired presentation of multimedia content exactly in the way specified; for advanced operations like retrieval and reuse, automatic composition, recommendation, and adaptation of content according to user interests, information needs, and technical infrastructure, valuable information about the semantics of content is lacking.

In parallel to research on the Semantic Web (Berners-Lee et al., 2001; Fensel, 2001), one can therefore observe a shift in paradigm towards a *semantic* modeling of multimedia content. The basic media of which multimedia content consists are supplemented with metadata describing these media and their semantic interrelationships. These media and descriptions are processed by stylesheets, search engines, or user agents providing advanced functionality on the content that can exceed mere hard-wired playback.

Current semantic multimedia modeling approaches, however, largely treat the content's basic media, the semantic description, and the functionality offered on the content as separate entities: the basic media of which multimedia content consists are typically stored on web or media servers; the semantic descriptions of these media are usually stored in databases or in dedicated files on web servers using formats like RDF (Lassila & Swick, 1999) or Topic Maps (ISO/IEC JTC 1/SC 34/WG 3, 2000); the functionality on the content is normally realized as servlets or stylesheets running in application servers or as dedicated software running at the clients such as user agents.

This inherent separation of media, semantic description, and functionality in semantic multimedia content modeling, however, hinders the realization of multimedia content sharing as well as collaborative applications which are gaining more and more importance, such as

the sharing of MP3 music files (Gnutella, n.d.) or learning materials (Nejdl et al., 2002) or the collaborative authoring and annotation of multimedia patient records (Grimson et al., 2001). The problem is that exchanging content today in such applications simply means exchanging single media files. An analogous exchange of semantically modeled multimedia content would have to include content descriptions and associated functionality, which are only coupled loosely to the media and usually exist on different kinds of servers potentially under control of different authorities, and which are thus not easily moveable.

In this chapter, we give an illustrated introduction to Enhanced Multimedia Meta Objects (Emmos), a semantic multimedia content modeling approach developed with collaborative and content sharing applications in mind. Essentially, an Emmo constitutes a self-contained piece of multimedia content that merges three of the content's aspects into a single object: the *media aspect*, i.e., the media which make up the multimedia content, the *semantic aspect* which describes the content, and the *functional aspect* by which an Emmo can offer meaningful operations on the content and its description that can be invoked and shared by applications. Emmos in their entirety – including media, content description, and functionality - can be *serialized* into bundles and are *versionable*: essential characteristics that enable their exchangeability in content sharing applications as well as the distributed construction and modification of Emmos in collaborative scenarios.

Furthermore, this chapter illustrates how we employed Emmos for two concrete collaborative and content sharing applications in the domains of cultural heritage and digital music archives.

The chapter is organized as follows: we begin with an overview of Emmos and show their difference to existing approaches for multimedia content modeling. We then introduce the conceptual model behind Emmos and outline a distributed Emmo container infrastructure for the storage, exchange, and collaborative construction of Emmos. We then apply Emmos for

the representation of multimedia content in two application scenarios. We conclude this paper with a summary and give an outlook to our current and future work.

BACKGROUND

In this section, we provide a basic understanding of the Emmo idea by means of an illustrating example. We show the uniqueness of this idea by relating Emmos to other approaches to multimedia content modeling in the field.

The Emmo Idea

The motivation for the development of the Emmo model is the desire to realize multimedia content sharing and collaborative applications on the basis of semantically modeled content but to avoid the limitations and difficulties of current semantic modeling approaches implied by their isolated treatment of media, semantic description, and content functionality. Following an abstract vision originally formulated by (Reich et al., 2000), the essential idea behind Emmos is to keep semantic description and functionality tied to the pieces of content to which they belong, thereby creating self-contained units of semantically modeled multimedia content easier to exchange in content sharing and collaborative application scenarios. An Emmo coalesces the basic media of which a piece of multimedia content consists (i.e., the content's *media aspect*), the semantic description of these media (i.e., the content's *semantic aspect*), and functionality on the content (i.e., the content's *functional aspect*) into a single serializeable and versionable object.

Figure 1 depicts a sketch of a simplified Emmo, which models a small multimedia photo album of a holiday trip of an imaginary couple Paul and Mary and their friend Peter. The bottom of the figure illustrates how Emmos address the *media aspect* of multimedia content. An Emmo acts as a container of the basic media of which the content that is represented by

the Emmo consists. In our example, the Emmo contains four JPEG images which constitute the different photographs of the album along with corresponding thumbnail images.

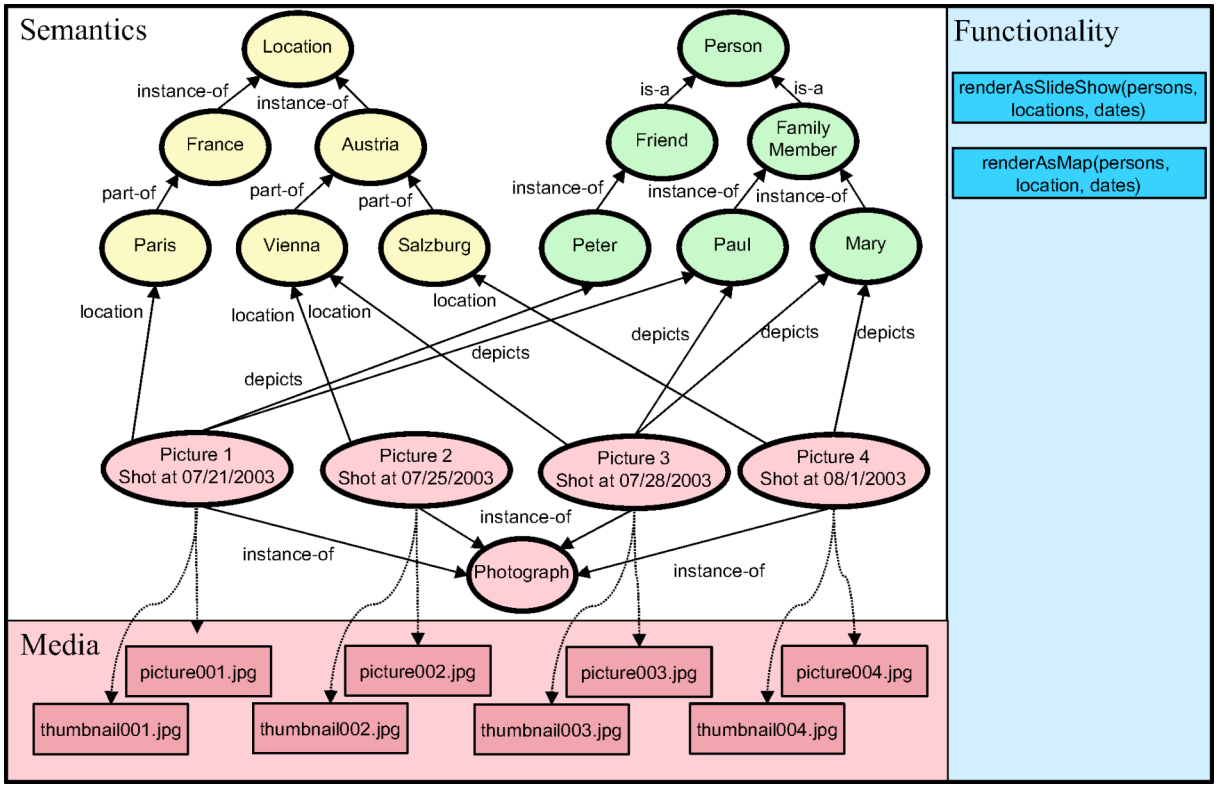

Fig. 1. Aspects of an Emmo

Media can be contained either by inclusion, i.e., raw media data is directly embedded within an Emmo, or by reference via an URI if embedding raw media data is not feasible because of the size of media data or the media is a live stream.

An Emmo further carries a semantic description of the basic media it contains and the associations between them. This *semantic aspect*, illustrated to the upper left of Figure 1, makes an Emmo a unit of knowledge about the multimedia content it represents.

For content description, Emmos apply an expressive concept graph-like data model similar to RDF and Topic Maps. In this graph model, the description of the content represented by an Emmo is not performed directly on the media that are contained in the Emmo; instead, the model abstracts from physical media making it possible to subsume several media objects which constitute only different physical manifestations of logically one and the same medium

under a single media node. This is a convenient way to capture alternative media. In Figure 1, for example, each media node `Picture 1` … `Picture 4` subsumes not only a photo but also its corresponding thumbnail image.

Apart from media, nodes can also represent abstract concepts. By associating an Emmo's media objects with such concepts, it is possible to create semantically rich descriptions of the multimedia content the Emmo represents. In Figure 1, for instance, it is expressed that the logical media nodes `Picture 1` … `Picture 4` constitute photos taken in Paris, Vienna, and Salzburg showing Peter and Paul, Paul and Mary, and Mary, respectively. The figure further indicates that nodes can be augmented with primitive attribute values for closer description: the pictures of the photo album are furnished with the dates at which they have been shot.

By associating concepts with each other, it is also possible to express domain knowledge within an Emmo. It is stated in our example that Peter, Paul, and Mary are Persons, that Paul and Mary are family members, that Peter is a friend, that Paris is located in France, and that Vienna and Salzburg are parts of Austria.

The Emmo model does not predefine the concepts, association types, and primitive attributes available for media description; these can be taken from arbitrary, domain-specific ontologies. While they thus constitute a very generic, flexible, and expressive approach to multimedia content modeling, Emmos are no ready-to-use formalism but require an agreed common ontology before they can be employed in an application.

Finally, Emmos also address the *functional aspect* of content. An Emmo can offer operations that can be invoked by applications in order to work with the content the Emmo represents in a meaningful manner. As shown to the top right of Figure 1, our example Emmo provides two operations supporting two different rendition options for the photo album, which are illustrated by the screenshots of Figure 2. As indicated by the left screenshot, the operation

`renderAsSlideshow()` might know how to - given a set of persons, locations, as well as time periods of interest - render the photo album as a classic slideshow on the basis of the contained pictures and their semantic description by generating an appropriate SMIL presentation. As indicated by the right screenshot, the operation `renderAsMap()` might also know how to – given the same data - render the photo album as a map with thumbnails pointing to the locations where photographs have been taken by constructing an SVG graph.

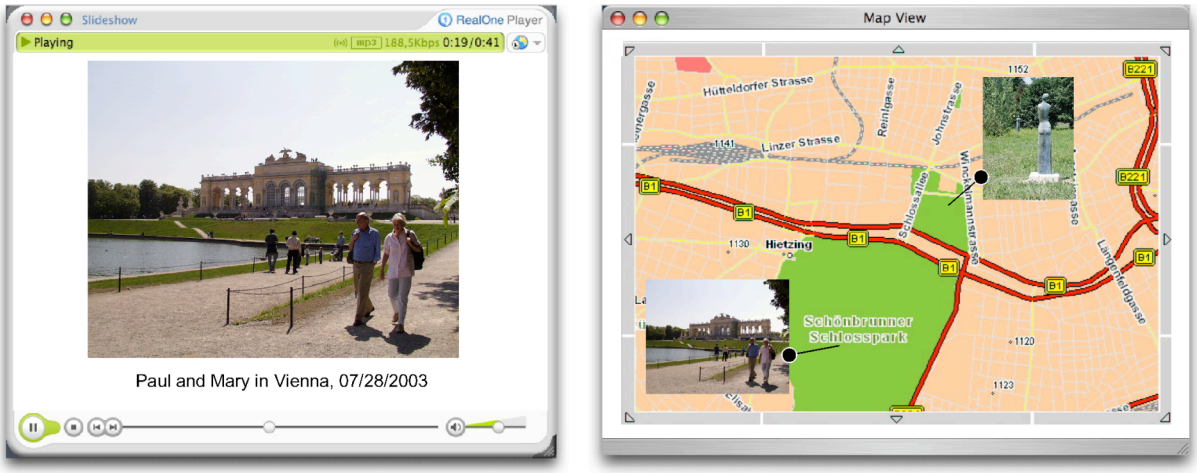

Fig. 2. Emmo Functionality

One may think of many further uses of operations. For example, operations could also be offered for rights clearance, displaying terms of usage, etc.

Emmos have further properties: an Emmo can be *serialized* and shared in its entirety in a distributed content sharing scenario including its contained media, the semantic description of these media, and its operations. In our example, this means that Paul can accord Peter the photo album Emmo as a whole - for instance, via email or a file-sharing peer-to-peer infrastructure - and Peter can do anything with the Emmo what Peter can also do, including invoking its operations.

Emmos also support *versioning*. Every constituent of an Emmo is versionable, an essential prerequisite for applications requiring the distributed and collaborative authoring of multimedia content. This means that Peter, having received the Emmo from Paul, can add his

own pictures to the photo album while Paul can still modify his local copy. Thereby, two concurrent versions of the Emmo are created. As the Emmo model is able to distinguish both versions, Paul can merge them into a final one when he receives Peter's changes.

Related Approaches

The fundamental idea underlying the concept of Emmos presented beforehand is that an Emmo constitutes an object unifying three different aspects of multimedia content, namely the media aspect, the semantic aspect, and the functional aspect. In the following, we fortify our claim that this idea is unique.

Interrelating basic media like single images and videos to form multimedia content is the task of multimedia document models. Recently, several standards for multimedia document models have emerged (Boll et al., 2000), such as HTML (Ragett et al., 1999), XHTML+SMIL (Newmann et al., 2002), HyTime (ISO/IEC JTC 1/SC 34/WG 3, 1997), MHEG-5 (ISO/IEC JTC 1/SC 29, 1997), MPEG-4 BIFS and XMT (Pereira & Ebrahimi, 2002), SMIL (Ayars et al., 2001), and SVG (Ferraiolo et al., 2003). Multimedia document models can be regarded as composite media formats that model the presentation of multimedia content by arranging basic media according to temporal, spatial, and interaction relationships. They thus mainly address the media aspect of multimedia content. Compared to Emmos, however, multimedia document models neither interrelate multimedia content according to semantic aspects nor do they allow to provide functionality on the content. They rely on external applications like presentation engines for content processing.

As a result of research concerning the Semantic Web, a variety of standards have appeared that can be used to model multimedia content by describing the information it conveys on a semantic level, such as RDF (Lassila & Swick, 1999; Brickley & Guha, 2002), Topic Maps (ISO/IEC JTC 1/SC 34/WG 3, 2000), MPEG-7 (especially MPEG-7's graph tools for the

description of content semantics (ISO/IEC JTC 1/SC 29/WG 11, 2001)), and Conceptual Graphs (ISO/JTC1/SC 32/WG 2, 2001). These standards clearly cover the semantic aspect of multimedia content. As they also offer means to address media within a description, they undoubtedly refer to the media aspect of multimedia content as well. Compared to Emmos, however, these approaches do not provide functionality on multimedia content. They rely on external software like database and knowledge base technology, search engines, user agents, etc. for the processing of content descriptions. Furthermore, media descriptions and the media described are separate entities - potentially scattered around different places on the Internet, created and maintained by different and unrelated authorities not necessarily aware of each other and not necessarily synchronized - whereas Emmos combine media and their semantic relationships into a single indivisible unit.

There exist several approaches that represent multimedia content by means of objects. Enterprise Media Beans (EMBs) (Baumeister, 2002) extend the Enterprise Java Beans (EJBs) architecture (Matena & Hapner, 1998) with predefined entity beans for the representation of basic media within enterprise applications. These come with rudimental access functionality but can be extended with arbitrary functionality using the inheritance mechanisms available to all EJBs. Though addressing the media and functional aspects of content, EMBs in comparison to Emmos are mainly concerned with single media content and not with multimedia content. Furthermore, EMBs do not offer any dedicated support for the semantic aspect of content.

Adlets (Chang & Znati, 2001) are objects that represent individual (not necessarily multimedia) documents. Adlets support a fixed set of predefined functionality which enables them to advertise themselves to other Adlets. They are thus content representations that address the media as well as the functional aspect. Different from Emmos, however, the

functionality supported by Adlets is limited to advertisement and there is no explicit modeling of the semantic aspect.

Tele-Action Objects (TAOs) (Chang et al., 1995) are object representations of multimedia content that encapsulate the basic media of which the content consists and interlink them with associations. Though TAOs thus address the media aspect of multimedia content in a way similar to Emmos, they do not adequately cover the semantic aspect of multimedia content: only a fixed set of 5 association types is supported mainly concerned with temporal and spatial relationships for presentation purposes. TAOs can further be augmented with functionality. Such functionality is, in contrast to the functionality of Emmos, automatically invoked as the result of system events and not explicitly invoked by applications.

Distributed Active Relationships (Daniel et al., 1998) define an object model based on the Warwick Framework (Lagoze et al., 1996). In the model, Digital Objects (DOs), which are interlinked with each other by semantic relationships, act as containers of metadata describing multimedia content. DOs thus do not address the media aspect of multimedia content but focus on the semantic aspect. The links between containers can be supplemented with arbitrary functionality. As a consequence, DOs take account of the functional aspect as well. Different from Emmos, however, the functionality is not explicitly invoked by applications but implicitly whenever an application traverses a link between two DOs.

ENHANCED MULTIMEDIA META OBJECTS

Having motivated and illustrated the basic idea behind them, this section semiformally introduces the conceptual model underlying Emmos using UML class diagrams. A formal definition of this model can be found in (Schellner et al., 2003). The discussion is oriented along the three aspects of multimedia content encompassed by Emmos: the media aspect, the semantic aspect, and the functional aspect.

Media Aspect

Addressing the media aspect of multimedia content, an Emmo encapsulates the basic media of which the content it represents is composed. Figure 3 presents the excerpt of the conceptual model which is responsible for this.

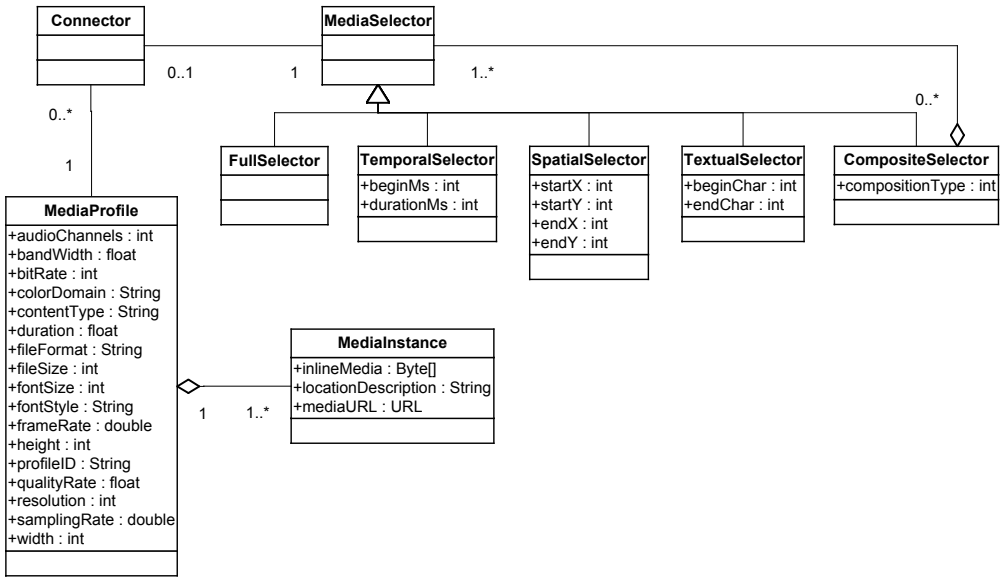

Fig. 3. Management of basic media in an Emmo

Closely following the MPEG-7 standard and its multimedia description tools (ISO/IEC JTC 1/SC 29/WG 11, 2001), basic media are modeled by *media profiles* (represented by the class `MediaProfile` in Figure 3) along with associated media instances (represented by the class `MediaInstance`). Media profiles hold low-level metadata describing physical characteristics of the media such as the storage format, file size, etc.; the media data itself is represented by *media instances*, each of which may directly embed the data in form of a byte array or, if that is not possible or feasible, address its storage location by means of an URI. Moreover, if a digital representation is not available, a textual location description can be specified, e.g. the location of analog tapes in some tape archive. Figure 3 further shows that a media profile can have more than one media instances. In this way, an Emmo can be provided with information about alternative storage locations of media.

Basic media represented by media profiles and media instances are attached to an Emmo by means of a *connector* (see class `Connector` in Figure 3). A connector not just addresses a basic medium via a media profile; it may also refer to a *media selector* (see base class `MediaSelector`) to address only a part of the medium. As indicated by the various subclasses of `MediaSelector`, it is possible to select media parts according to simple textual, spatial, temporal and textual criteria, as well as an arbitrary combination of these criteria (see class `CompositeSelector`). It is thus possible to address the upper right part of a scene in a digital video starting from second 10 and lasting until second 30 within an Emmo without having to extract that scene and to put it into a separate media file using a video editing tool.

Semantic Aspect

Out of the basic media which it contains, an Emmo forges a piece of semantically modeled multimedia content by describing these media and their semantic interrelationships. The class diagram of Figure 4 gives an overview over the part of the Emmo model that provides these semantic descriptions. As one can see, the basic building blocks of the semantic descriptions, the so-called *entities*, are subsumed under the common base class `Entity`. The Emmo model distinguishes four kinds of entities, namely *logical media parts*, *associations*, *ontology objects*, and *Emmos* themselves, represented by according subclasses. These four kinds of entities have a common nature but each extends the abstract notion of an entity with additional characteristic features.

Figure 5 depicts the characteristics that are common to all kinds of entities. Each entity is globally and uniquely identified by its OID, realized by means of a universal unique identifier (UUID) (Leach, 1998) which can be easily created even in distributed scenarios. To enhance human readability and usability, each entity is further augmented with additional attributes

like a name and a textual description. Moreover each entity holds information about its creator and its creation and modification date.

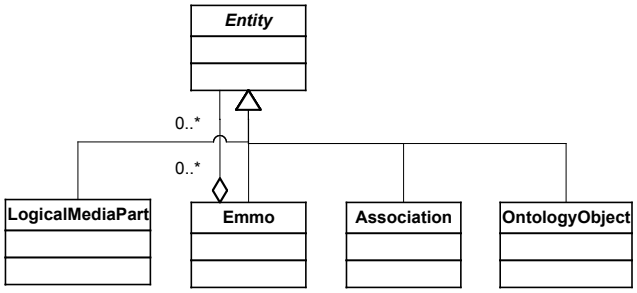

Fig. 4. The semantic aspect of Emmos

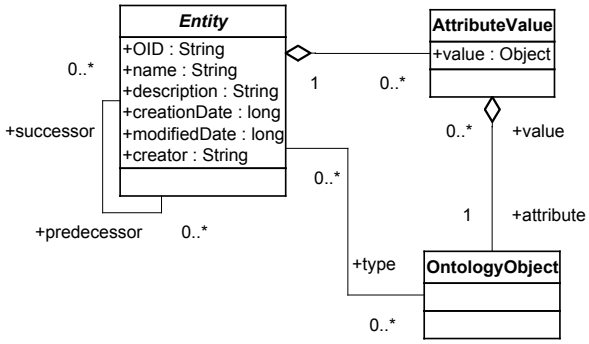

Fig. 5. Entity details

Figure 5 furthermore expresses that entities may receive an arbitrary number of *types*. A type is a concept taken from an ontology and represented by an ontology object in the model. Types thus constitute entities themselves. By attaching types, an entity gets meaning and is classified in an application-dependent ontology. As mentioned before, the Emmo model does not come with a predefined set of ontology objects but instead relies on applications to agree on common ontology before the Emmo model can be used.

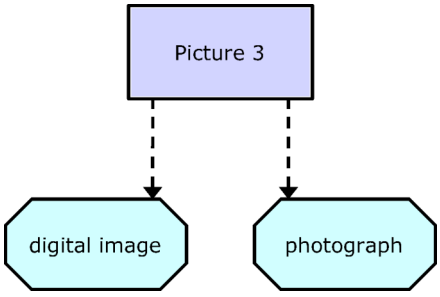

Fig. 6. An entity with its types

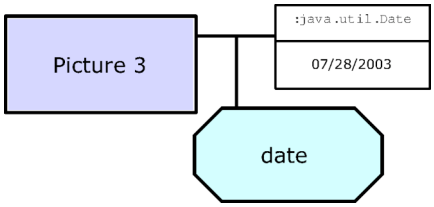

Fig. 7. An entity with an attribute value

In the example of Figure 6, the entity `Picture 3` of kind logical media part (depicted as a rectangle), which represents the third picture of our example photo album of the holiday trip introduced in the previous section, is an instantiation of the concepts “photograph” and “digital image”, represented by the ontology objects `photograph` and `digital image` (each pictured by an octagon) respectively. The type relationships are indicated by dashed arrows.

For further description, the Emmo model also allows to attach arbitrary *attribute values* to entities (expressed by the class of the same name in the class diagram of Figure 5). Attribute values are simple attribute-value pairs, with the attributes being a concept of an application-dependent ontology represented by an ontology object entity and the value being an arbitrary object suiting the type of the value. The rationale behind representing attributes by concepts of an ontology and not just by mere string identifiers is that this allows to express constraints on the usage of attributes within the ontology, e.g., to which entity types attributes are applicable.

Figure 7 gives an example of attribute values. In the figure, it is stated that the third picture of the photo album has been taken at July 28th 2003 by attaching an attribute value "date=07/28/2003" to the entity `Picture 3` representing that picture. The attribute "date" is modeled by the ontology object `date` and the value "07/28/2003" is captured by an object of a suitable date class (represented using the UML object notation).

As an essential prerequisite for the realization of distributed, collaborative multimedia applications in which multimedia content is simultaneously authored and annotated by different persons at different locations, the Emmo model provides intrinsic support for versioning. The class diagram of Figure 5 states that every entity is versionable and can have an arbitrary number of predecessor and successor versions, all of which have to be entities of the same kind as the original entity. Treating an entity's versions as entities on their own has several benefits: on the one hand, entities constituting versions of other entities have their own globally unique OID. Hence, different versions concurrently derived from one and the same entity at different sites can easily be distinguished without synchronization effort. On the other hand, different versions of an entity can be interrelated just like any other entities allowing to establish comparative relationships between entity versions.

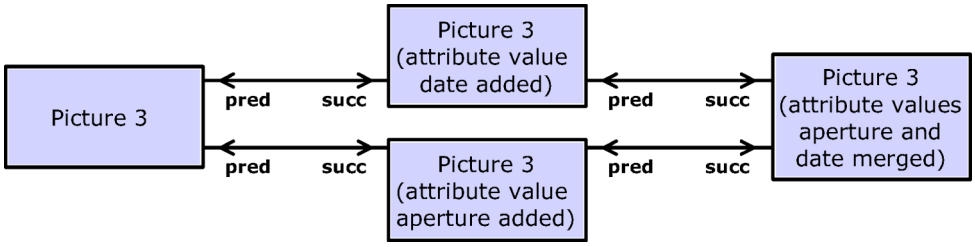

Fig. 8. Versioning of an entity

Figure 8 exemplifies a possible versioning of our example entity `Picture 3`. The original version of this logical part is depicted to the left of the figure. As expressed by the special arrows indicating the predecessor (`pred`) and the successor (`succ`) relationship between different versions of the same entity, two different successor versions of this original version were created, possibly by two different people at two different locations. One version augments the logical media part with a `date` attribute value to denote the creation date of the picture whereas the other provides an attribute value describing the aperture with which the picture was taken. Finally, as shown by the logical media part at the right side of the figure, these two versions were merged again into a fourth that now holds both attribute values. Having explained the common characteristics shared by all entities, we are now able to introduce the peculiarities of the four concrete kinds of entities: logical media parts, ontology objects, associations, and Emmos.

Logical Media Parts

Logical media parts are entities that form the bridge between the semantic aspect and the media aspect of an Emmo. A logical media part represents a basic medium of which multimedia content consists on a logical level for semantic description, thereby providing an abstraction from the physical manifestation of the medium. According to the class diagram of Figure 9, logical media parts can refer to an arbitrary number of connectors - which we already know from our description of the media aspect of Emmos - permitting to logically subsume alternative media profiles and instances representing different media files in possibly

different formats in possibly different storage locations under a common logical media part. The ID of the default profile to use is identified via the attribute `masterProfileID`. Since logical media parts do not need to have connectors associated with them, it is also possible to refer to media within Emmos which do not have a physical manifestation.

Ontology Objects

Ontology objects are entities that represent concepts of an ontology. We have already described how ontology objects are used to define an entity types and to augment entities with attribute values. By relating entities such as logical media parts to ontology objects, they can be given a meaning. As it can be seen from the class diagram of Figure 10, the Emmo model distinguishes two kinds of ontology objects represented by two subclasses of `OntologyObject`: `Concept` and `ConceptRef`. Whereas an instance of `Concept` serves to represent a concept of an ontology that is fully captured within the Emmo model, `ConceptRef` allows to reference concepts of ontologies specified in external ontology languages such as RDF Schema (Brickley & Guha, 2002). The latter is a pragmatic tribute to the fact that we have not developed an ontology language for Emmos yet and therefore rely on external languages for this purpose. References to concepts of external ontologies additionally need a special ID (`objOID`) uniquely identifying the external concept referenced and a label indicating the format of the ontology (`ontStandard`), e.g., "RDF Schema".

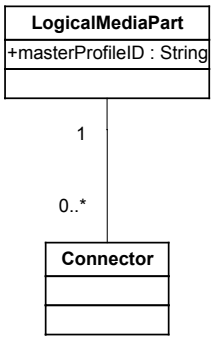

Fig. 9. Logical media parts

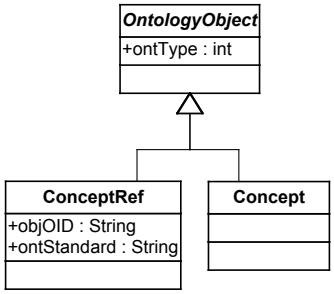

Fig. 10. Ontology objects

Associations

Associations are entities that establish binary directed relationships between entities, allowing to create complex and detailed descriptions of the multimedia content represented by the Emmo. As one can see from Figure 11, each association has exactly one *source entity* and one *target entity*. The kind of semantic relationship represented by an association is defined by the association's *type* which is - like the types of other entities - an ontology object representing the concept that captures the type in an ontology. Different from other entities, however, an association is only permitted to have one type as it can express only a single kind of relationship.

Since associations are first-class entities, they can take part as sources or targets in other associations like any other entities. This feature permits the creation of very complex content descriptions, as it facilitates the reification of statements ("statements about statements") within the Emmo model.

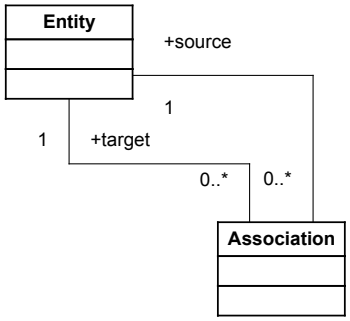

Fig. 11. Association

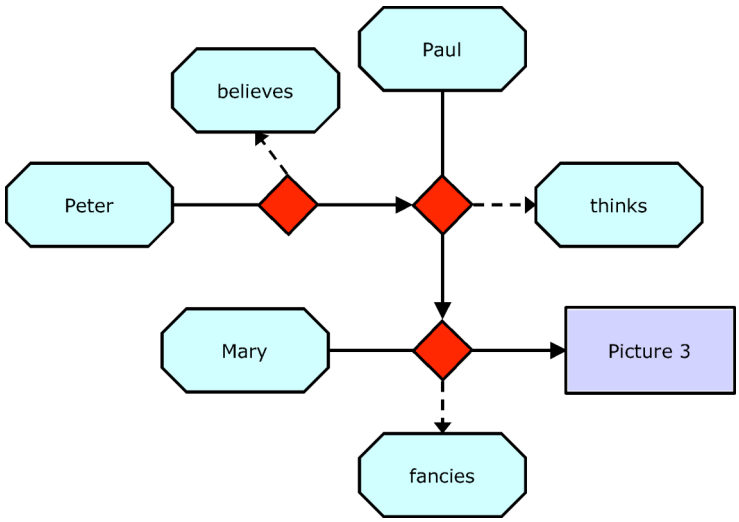

Fig. 12. Reification

Figure 12 demonstrates how reification can be expressed. In the figure, associations are symbolized by a diamond shape, with solid arrows indicating the source and target of an association and dashed arrows indicating the association type. The example shown in this figure wants to express that "Peter believes that Paul thinks that Mary fancies Picture 3". The

statement “Mary fancies Picture 3” is represented at the bottom of the figure by an association of type `fancies` that connects the ontology object `Mary` with the logical media part `Picture3`. Moreover, this association acts as target for another association having the type `thinks` and the source entity `Paul`, thereby making a statement about the statement “Mary fancies Picture 3”. This reification is then further enhanced by attaching another statement to obtain the desired message.

Emmos

Emmos themselves, finally, constitute the fourth kind of entities. An Emmo is basically a container that encapsulates arbitrary entities to form a semantically modeled piece of multimedia content (see the aggregation between the classes `Emmo` and `Entity` in the introductory outline of the model in Figure 4). As one and the same entity can be contained in more than one Emmo, it is possible to encapsulate different, context-dependent, and even contradicting views onto the same content within different Emmos; as Emmos are first-class entities, they can be contained within other Emmos and take part in associations therein, allowing to build arbitrarily nested Emmo structures for the logical organization of multimedia content. These are important characteristics especially useful for the authoring process, as they facilitate reuse of existing Emmos and the content they represent.

Figure 13 shows an example where a particular Emmo encapsulates another. In the figure, Emmos are graphically shown as ellipses. The example depicts an Emmo modeling a private photo gallery that up to the moment holds only a single photo album (again modeled by an Emmo), namely the photo album of the journey to Europe we used as a motivating example in the section illustrating the Emmo idea. Via an association, this album is classified as “vacation“ within the photo gallery. In the course of time, the photo gallery might become filled with additional Emmos representing further photo albums, e.g., one that keeps the

photos of a summer vacation in Spain. These Emmos can be related to each other. For example, an association might express that the journey to Europe took place before the summer vacation in Spain.

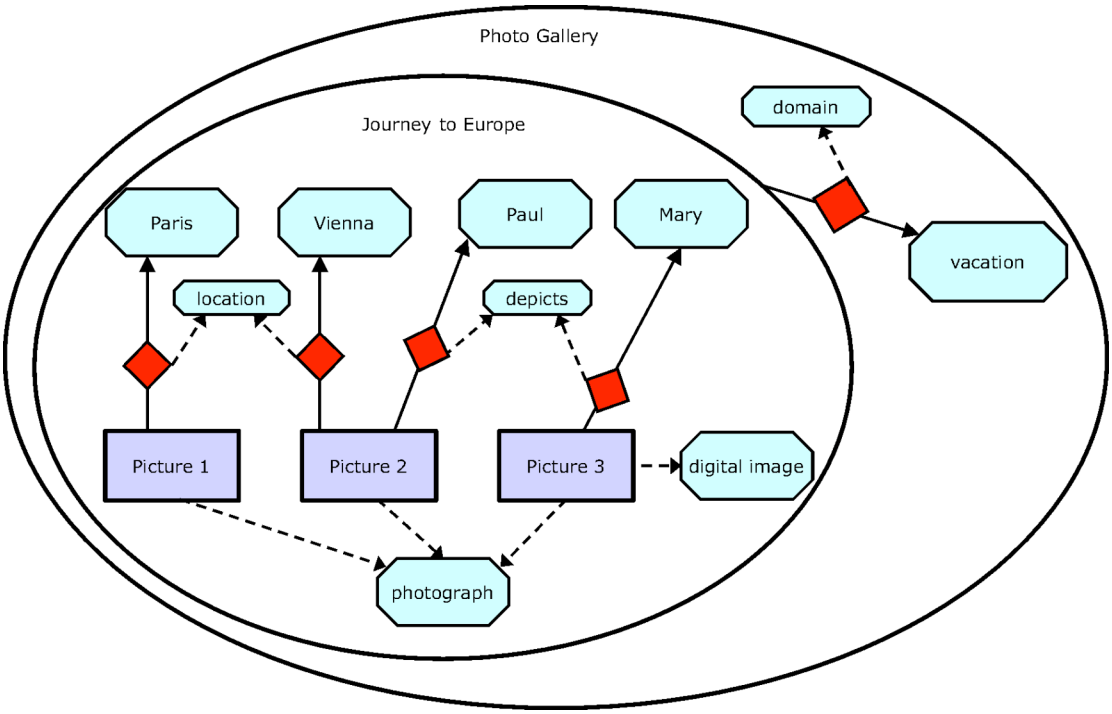

Fig. 13. Nested Emmos

Functional Aspect

Emmos also address the functional aspect of multimedia content. Emmos may offer *operations* that realize arbitrary content-specific functionality which makes use of the media and descriptions provided with the media and semantic aspects of an Emmo and which can be invoked by applications working with content. The class diagram of Figure 14 shows how this is realized in the model. As expressed in the diagram, an Emmo may aggregate an arbitrary number of operations represented by the class of the same name. Each operation has a *designator*, i.e., a name that describes its functionality, which is represented by an ontology object. Similar to attributes, the motivation behind using concepts of an ontology as operation designators instead of simple string identifiers is that this allows to express restrictions on the

usage of operations within an ontology, for example the types of Emmos for which an operation is available, the types of the expected input parameters, etc.

The functionality of an operation is provided by a dedicated *implementation class* whose name is captured by an operation's `implClassName` attribute to permit the dynamic instantiation of the implementation class at runtime. There are not many restrictions for such an implementation class: the Emmo model merely demands that an implementation class realizes the `OperationImpl` interface. `OperationImpl` enforces the implementation of a single method only, namely the method `execute()` which expects the Emmo on which an operation is executed as its first parameter followed by a vector of arbitrary operation-dependent parameter objects. `execute()` performs the desired functionality and, as a result, may return an arbitrary object.

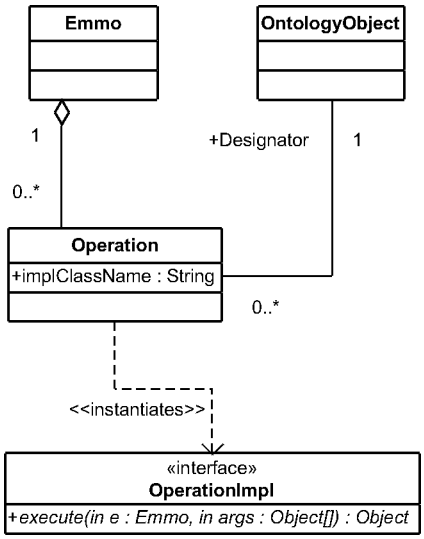

Fig. 14. Emmo's functionality

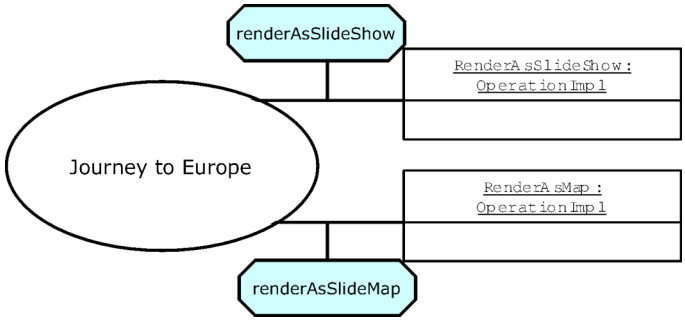

Fig. 15. Example of Emmo operations

Figure 15 once more depicts the Emmo modeling the photo album of the journey to Europe that we already know from Figure 13, but this time enriched with the two operations already envisioned in the second section: one that traverses the semantic description of the album returns a SMIL presentation that renders the album as a slide show and another that returns an SVG presentation that renders the same album as a map. For both operations, two

implementation classes are provided that are attached to the Emmo and differentiated via their designators `renderAsSlideShow` and `renderAsMap`.

THE EMMO CONTAINER INFRASTRUCTURE

As an elementary foundation for the sharing and collaborative authoring of pieces of semantically modeled multimedia content on the basis of the Emmo model, we have implemented a distributed Emmo container infrastructure. Figure 16 provides an overview of this infrastructure, which we are going to describe in more detail in the following.

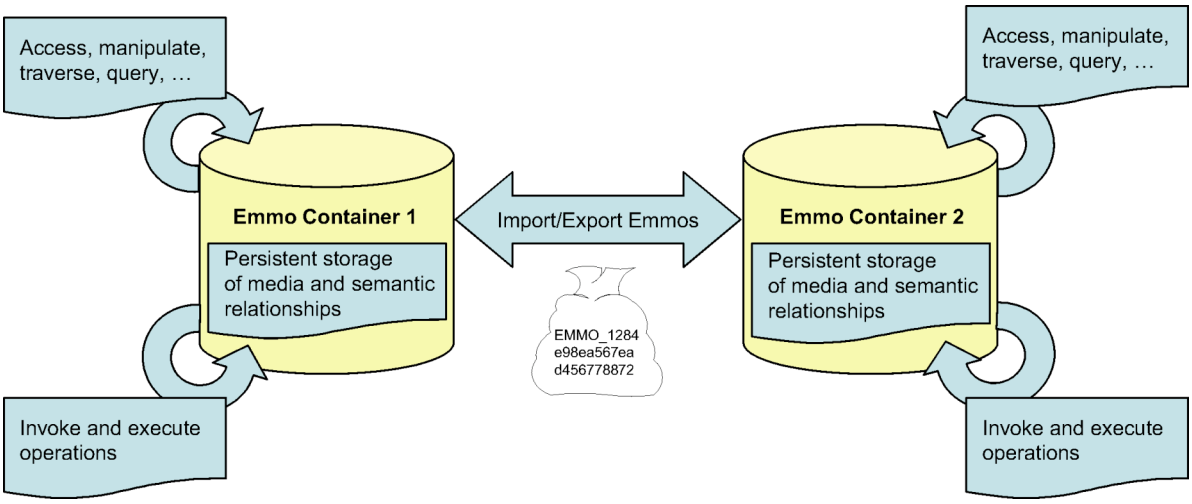

Fig. 16. Emmo container infrastructure

Basically, an Emmo container provides a space where Emmos "live". Its main purpose is the management and persistent storage of Emmos. An Emmo container provides application programming interfaces that permit applications to fine-grainedly access, manipulate, traverse, and query the Emmos it stores. This includes the media aspect of an Emmo with its media profiles and instances, the semantic aspect with all its descriptional entities such as logical media parts, ontology objects, other Emmos, and associations, as well as the versioning relationships between those entities. Moreover, an Emmo container offers an interface to invoke and execute an Emmo's operations giving access to the functional aspect of an Emmo.

Emmo containers are not intended as a centralized infrastructure with a single Emmo container running at a server (although this is possible). Instead, it is intended to establish a decentralized infrastructure with Emmo containers of different scales and sizes running at each site that works with Emmos. Such a decentralized Emmo management naturally reflects the nature of content sharing and collaborative multimedia applications.

The decentralized approach has two implications. The first implication is that *platform independence* and *scalability* are important in order to support Emmo containers at potentially very heterogeneous sites ranging from home users to large multimedia content publishers with different operating systems, capabilities, and requirements.

For these reasons, we have implemented the Emmo containers in Java, employing the object-oriented DBMS ObjectStore for persistent storage. By Java, we obtain platform independence; by ObjectStore, we obtain scalability as there not just exists a full-fledged database server implementation suitable for larger content providers but also a code-compatible file-based in-process variant named PSEPro better suiting the limited needs of home users. It would have been possible to use a similarly scalable relational DBMS for persistent storage as well; we opted for an object-oriented DBMS, however, because of these systems' suitability for handling complex graph structures like Emmos.

The second implication of a decentralized infrastructure is that Emmos must be *transferable* between the different Emmo containers operated by users that want to share or collaboratively work on content. This requires Emmo containers to be able to completely export Emmos into bundles encompassing their media, semantic, and functional aspects, and to import Emmos from such bundles, which is explained in more detail in the following two subsections.

In the current state of implementation, Emmo containers are rather isolated components, requiring applications to explicitly initiate the import and export of Emmos and to manually transport Emmo bundles between different Emmo containers themselves. We are building a

peer-to-peer infrastructure around Emmo containers that permits the transparent search for and transfer of Emmos across different containers.

Exporting Emmos

An Emmo container can export an Emmo into a bundle whose overall structure is illustrated by Figure 17.

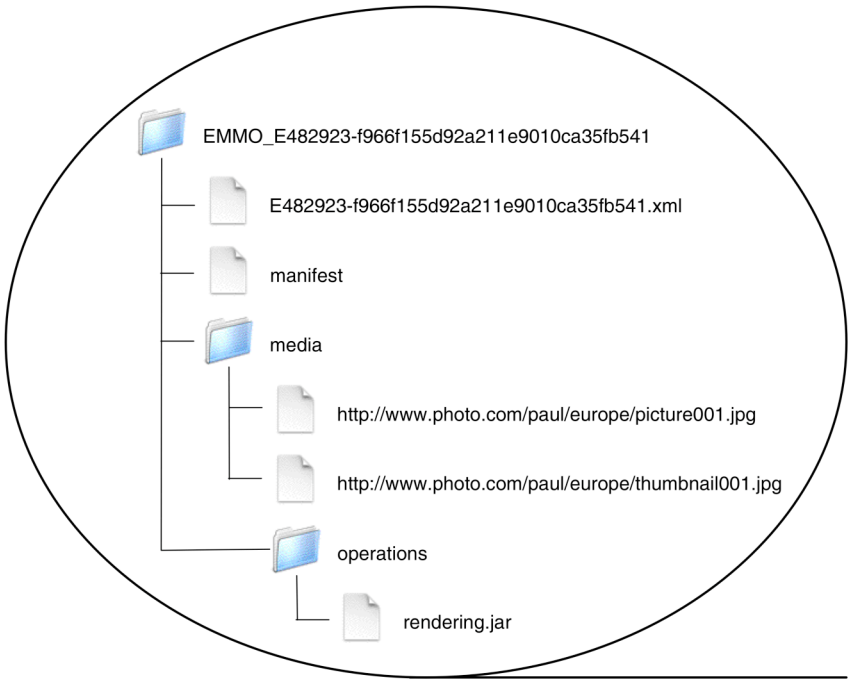

Fig. 17. Structure of an Emmo bundle

The bundle is basically a ZIP archive which captures all three aspects of an Emmo: the media aspect is captured by the bundle's `media` folder. The basic media files of which the multimedia content modeled by the Emmo consists are stored in this folder.

The semantic aspect is captured by a central XML file whose name is given the OID of the bundled Emmo. This XML file captures the semantic structure of the Emmo, thus describing all of the Emmo's entities, the associations between them, the versioning relationships, etc. Figure 18 shows a fragment of such an XML file. It is divided into a `<components>` section declaring all entities and media profiles relevant for the current Emmo and a

`<links>` section capturing all kinds of relationships between these entities and media profiles, such as types, associations, etc.

```
<?xml version="1.0" encoding="UTF-16"?>
<!-- Document created by org.cultos.storage.mdwb.exporter.MdwbXMLExporter -->
<emmo xmlns="http://www.cultos.org/emmos" xmlns:mpeg7="http://www.mpeg7.org/2001/
MPEG-7_Schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
schemaLocation="http://www.cultos.org/emmos http://www.cultos.org/emmos/XML/
emmo.xsd">
   <components>
      <entities>
         <entity xsi:type="LogicalMediaPart" mode="strong">
            <oid>E1a8d252-f8f2e098bb-3c7cb04afdbd1144ba4d1ea866d93db2</oid>
            <name>Beethoven's 5th Symphony</name>
            <creationDate>19 November 2003 08:24:54  CET</creationDate>
            <modifiedDate>19 November 2003 08:24:55  CET</modifiedDate>
         </entity>
         <entity xsi:type="Concept" mode="weak">
            <oid>E1a8d252-f8f2d7a8a5-3c7cb04afdbd1144ba4d1ea866d93db2</oid>
            <name>Classical music</name>
            <creationDate>19 November 2003 08:15:09 CET</creationDate>
            <modifiedDate>19 November 2003 08:15:09 CET</modifiedDate>
            <ontologyType>0</ontologyType>
         </entity>
         .....
      </entities>
      <mediaProfiles/>
   </components>
   <links>
      <types>
         <typeLink entity="E1a8d252-f8f2e095ed-3c7cb04afdbd1144ba4d1ea866d93db2"
                   type="E1a8d252-f8f2e0980f-3c7cb04afdbd1144ba4d1ea866d93db2"/>
         <typeLink entity="E1a8d252-f8f2e095fc-3c7cb04afdbd1144ba4d1ea866d93db2"
                   type="E1a8d252-f8f2e098ef-3c7cb04afdbd1144ba4d1ea866d93db2"/>
         <typeLink entity="E1a8d252-f8f2e098bb-3c7cb04afdbd1144ba4d1ea866d93db2"
                   type="E1a8d252-f8f2d7a8a5-3c7cb04afdbd1144ba4d1ea866d93db2"/>
      </types>
      <attributeValues/>
      <associations>
         <assoLink association="E1a8d252-f8f2e0981a-3c7cb04afdbd1144ba4d1ea866d93db2"
                   sourceEntity="E1a8d252-f8f2e095fc-3c7cb04afdbd1144ba4d1ea866d93db2"
                   targetEntity="E1a8d252-f8f2e098bb-3c7cb04afdbd1144ba4d1ea866d93db2"/>
      </associations>
      <connectors/>
      <predVersions/>
      <succVersions/>
      <encapsulations/>
   </links>
</emmo>
```

Fig. 18. Emmo XML representation

The functional aspect of an Emmo is captured by the bundle's `operations` folder in which the binary code of the Emmo's operations is stored. Here, our choice for Java as the implementation language for Emmo containers comes handy again, as it allows us to transfer operations in form of JAR files with platform-independent bytecode even between heterogeneous platforms.

The export functionality can react to different application needs by offering several export variants: an Emmo can be exported with or without media included in the bundle, one can choose whether to also include media that are only referenced by URIs, the predecessor and successor versions of the contained entities can either be added to the bundle or omitted, and it can be decided whether to recursively export Emmos contained within an exported Emmo. The particular export variants chosen are recorded in the bundle's `manifest` file.

In order to implement these different export variants, an Emmo container distinguishes three different *modes* of how entities can be placed in a bundle:

- The *strong* mode is the normal mode for an entity. The bundle holds all information about an entity including its types, attribute values, immediate predecessor and successor versions, media profiles (in case of a logical media part), contained entities (in case of an Emmo), etc.
- The *hollow* mode is applicable to Emmos only. The hollow mode indicates that the bundle holds all information about an Emmo except the entities it contains. The hollow mode appears in bundles where it was chosen not to recursively export encapsulated Emmos. In this case, encapsulated Emmos receive the hollow mode; the entities encapsulated by those Emmos are excluded from the export.
- The *weak* mode indicates that the bundle contains only basic information about an entity, such as its OID, name, and description but no types, attribute values, etc. Weak mode entities appear in bundles that have been exported without versioning information. In this case, the immediate predecessor and successor versions of exported entities are placed into the bundle in weak mode; indirect predecessor and successor versions are excluded from the export.

The particular mode of an entity within a bundle is marked with the `mode` attribute in the entity's declaration in the bundle's XML file (see again Figure 18).

Importing Emmos

When importing an Emmo bundle exported in the way described in the previous subsection, an Emmo container essentially inserts all media files, entities, and operations included in the bundle into its local database. In order to avoid duplicates, the container checks whether an entity with the same OID or whether a media file or JAR file already exists in the local database before insertion. In such a case, the basic strategy of the importing container is that the local copy prevails.

However, the different export variants for Emmos and the different modes in which entities might occur in a bundle as well as the fact that in a collaborative scenario Emmos might have been concurrently modified without creating new versions of entities demand a more sophisticated handling of duplicate entities on the basis of a timestamp protocol. Depending on the modes of two entities with the same OID in the bundle and the local database and the timestamps of both entities essentially the following treatment is applied:

- A greater mode (weak < hollow < strong) in combination with a more recent timestamp always wins. Thus, if the local entity has a greater mode and a newer timestamp, it prevails and the entity in the bundle is ignored. Similarly, if the local entity has a lesser mode and an older timestamp, the entity in the bundle completely replaces the local entity in the database.
- If the local entity has a more recent timestamp but a lesser mode, additional data available for the entity in the bundle (entity types, attribute values, predecessor or successor versions, encapsulated entities in case of Emmos, or media profiles in case of logical media parts) complements the data of the local entity thereby raising its mode.
- In case of same modes but a more recent timestamp of the entity in the bundle, the entity in the bundle completely replaces the local entity in the database.

- In case of same modes but a more recent timestamp of the entity in the local database, the entity in the database prevails and the entity in the bundle is ignored.

APPLICATIONS

Having introduced and described the Emmo approach to semantic multimedia content modeling and the Emmo container infrastructure, this section illustrates how these concepts have been practically applied in two concrete multimedia content sharing and collaborative applications. The first application named CULTOS is in the domain of cultural heritage and the second application introduces a semantic jukebox.

CULTOS

CULTOS is an EU-funded project carried out from 2001 to 2003 with 11 partners from EU-countries, and Israel[1]. It has been the task of CULTOS to develop a multimedia collaboration platform for authoring, managing, retrieving, and exchanging *Intertextual Threads* (ITTs) (Benari et al., 2002; Schellner et al., 2003) - knowledge structures that semantically interrelate and compare cultural artifacts such as literature, movies, artworks, etc. This platform enables the community of intertextual studies to create and exchange multimedia-enriched pieces of cultural knowledge that incorporate the community's different cultural backgrounds - an important contribution to the preservation of European cultural heritage.

ITTs are basically graph structures that describe semantic relationships between cultural artifacts. They can take a variety of forms, ranging from spiders over centipedes to associative maps, like the one shown in Figure 19.

The example ITT depicted highlights several relationships of the poem "The Fall" by Tuvia Ribner to other works of art. It states that the poem makes reference to the 3rd book of Ovid's

[1] See http://www.cultos.org for more details on the project.

“Metamorphoses” and that the poem is an ekphrasis of the painting “Icarus' Fall” of the famous Dutch painter Breugel.

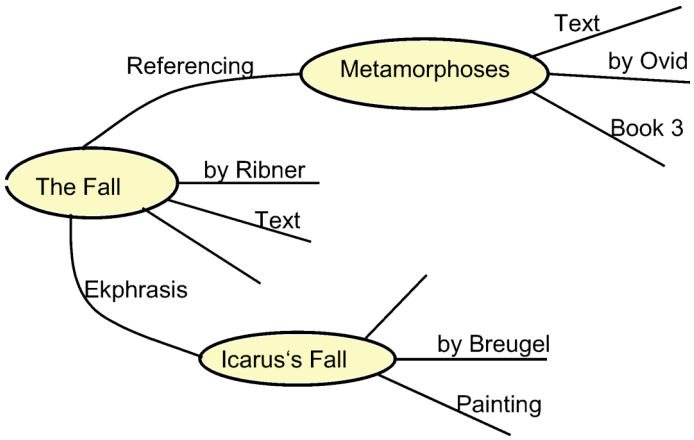

Fig. 19. Simple intertextual thread

The graphical representation of an ITT bears strong resemblance to well-known techniques for knowledge representation such as concept graphs or semantic nets, although it lacks their formal rigidity. ITTs nevertheless get very complex, as they commonly make use of constructs such as *encapsulation* and *reification* of statements that are challenging from the perspective of knowledge representation.

Encapsulation is intrinsic to ITTs because intertextual studies are no exact sciences. Certainly, the cultural and personal context of a researcher affects the kind of relationships between pieces of literature he discovers and are of value to him. As such different views onto a single subject are highly interesting to intertextual studies, ITTs themselves can be relevant subjects of discourse and thus be contained as first-class artifacts within other ITTs. Figure 20 illustrates this point with a more complex ITT that interrelates two ITTs manifesting two different views on Ribner’s poem as opposed representations.

Reification of statements is also frequently occurring within ITTs. Since experts in intertextual studies extensively base their position on the position of other researchers, statements about statements are common practice within ITTs. In the ITT of Figure 20, for

instance, it is expressed by reification that the statement describing the two depicted ITTs as opposed representation is only the opinion of a certain researcher B. Zoa.

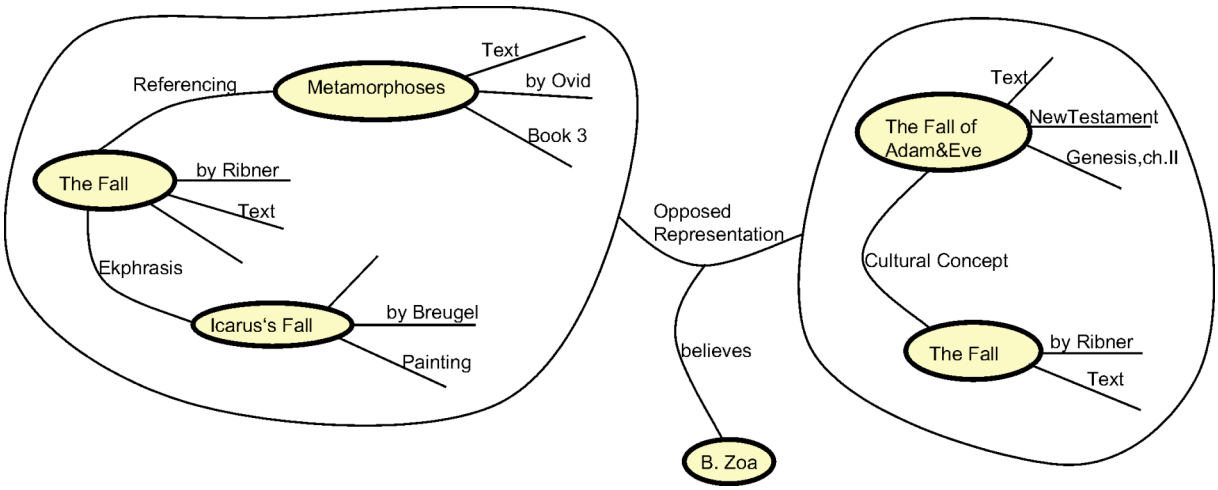

Fig. 20. Complex intertextual thread

Given these characteristics of ITTs, we have found that Emmos are very well suited for their representation in the multimedia collaboration platform for intertextual studies that is envisioned by CULTOS. Firstly, the semantic aspect of Emmos offers sufficient expressiveness to capture ITTs. Figure 21 shows how the complex ITT of Figure 20 could be represented using Emmos. Due to the fact that associations as well as Emmos themselves are first-class entities, it is even possible to cope with reification of statements as well as with encapsulation of ITTs.

Secondly, the media aspect of Emmos allows researchers to enrich ITTs that so far expressed interrelationships between cultural artefacts on an abstract level with digital media about these artefacts, such as a JPEG image showing Breugel's painting Icarus' Fall. The ability to consume these media while browsing an ITT certainly enhances the comprehension of the ITT and the relationships described therein.

Thirdly, with the functional aspect of Emmos, functionality can be attached to ITTs. For instance, an Emmo representing an ITT in CULTOS offers operations to render itself in an HTML-based hypermedia view.

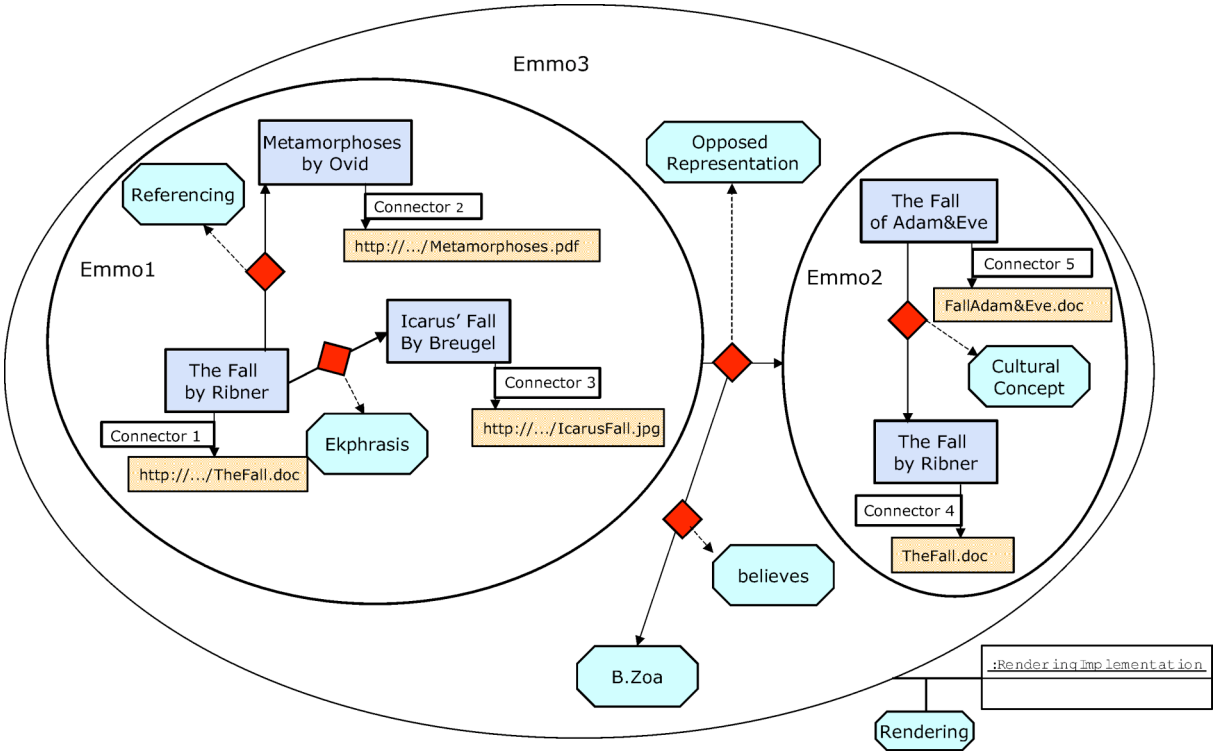

Fig. 21. Emmo representing an ITT

Additionally, our Emmo container infrastructure outlined in the previous section provides a suitable foundation for the realization of the CULTOS platform. Their ability to persistently store Emmos as well as their interfaces which enable applications to fine-grainedly traverse and manipulate the stored Emmos and invoke their operations make Emmo containers an ideal ground for the authoring and browsing applications for ITTs that had to be implemented in the CULTOS project. Figure 22 gives a screenshot of the authoring tool for ITTs that has been developed in the CULTOS project which runs on top of an Emmo container. Moreover, their decentralized approach allows the setup of independent Emmo containers at the sites of different researchers; their ability to import and export Emmos with all the aspects they cover facilitates the exchange of ITTs, including the media by which they are enriched as well as the functionality they offer. This enables researchers to share and collaboratively work on ITTs in order to discover and establish new links between artworks as well as different personal and cultural viewpoints thereby paving the way to novel insights to a subject. The profound versioning within the Emmo model further alleviates this kind of collaboration,

allowing researchers to concurrently create different versions of an ITT at different sites, to merge these versions, as well as to highlight difference between these versions.

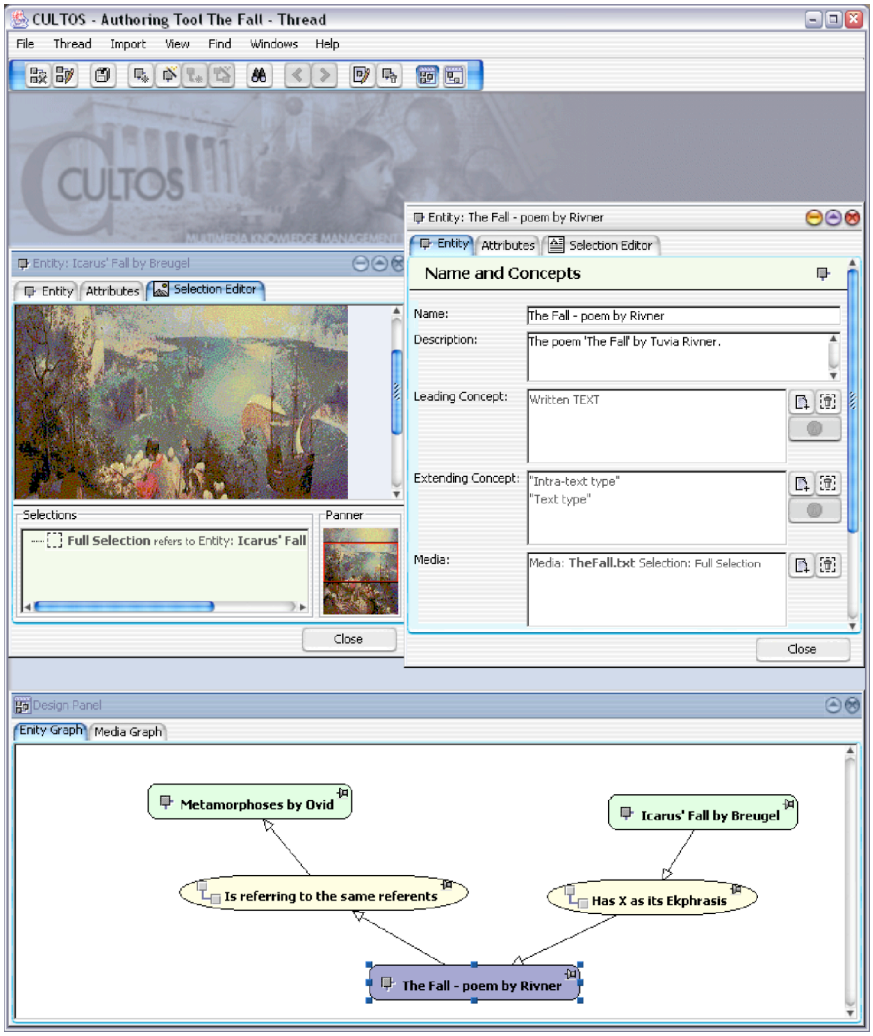

Fig. 22. CULTOS authoring tool for ITTs

Semantic Jukebox

One of the most prominent (albeit legally disputed) multimedia content sharing applications is the sharing of MP3 music files. Using peer-to-peer file sharing infrastructures such as Gnutella, many users gather large song libraries on their home PCs which they typically manage with one of the many jukebox programs available, such as Apple's iTunes (iTunes, n.d.). The increasing use of ID3 tags (ID3v2, n.d.) - optional free text attributes capturing

metadata like the interpreter, title, and the genre of a song - within MP3 files for song description alleviates the management of such libraries.

Nevertheless, ID3-based song management quickly reaches its limitations. While ID3 tags enable jukeboxes to offer reasonably effective search functionality for songs (provided the authors of ID3 descriptions spell the names of interprets, albums, and genres consistently), more advanced access paths to song libraries are difficult to realize. Apart from other songs of the same band or genre, for instance, it is difficult to find songs similar to the one that is currently playing. In this regard, it would also be interesting to be able to navigate to other bands in which artists of the current band played as well or with which the current band appeared on stage together. But such background knowledge cannot be captured with ID3 tags.

Using Emmos and the Emmo container infrastructure, we have implemented a prototype of a *semantic* jukebox that considers background knowledge about music. The experience we have gained from this prototype shows that the Emmo model is well-suited to represent knowledge-enriched pieces of music in a music sharing scenario. Figure 23 gives a sketch of such a music Emmo which holds some knowledge about the song “Round Midnight”.

Its media aspect enables the depicted Emmo to act as a container of MP3 music files. In our example, this is a single MP3 file with the song “Round Midnight” that is connected as a media profile to the logical media part `Round Midnight` in the center of the figure.

The Emmo’s semantic aspect allows us to express rich background knowledge about music files. For this purpose, we have developed a basic ontology for the music domain featuring concepts such as “Artist”, “Performance”, “Composition”, and “Record” that all appear as ontology objects in the figure. The ontology also features various association types which allow us to express that “Round Midnight” was composed by Thelonious Monk and the particular performance by Miles Davis can be found on the record “Round about Midnight”.

The ontology also defines attributes for expressing temporal information like the issue date of a record.

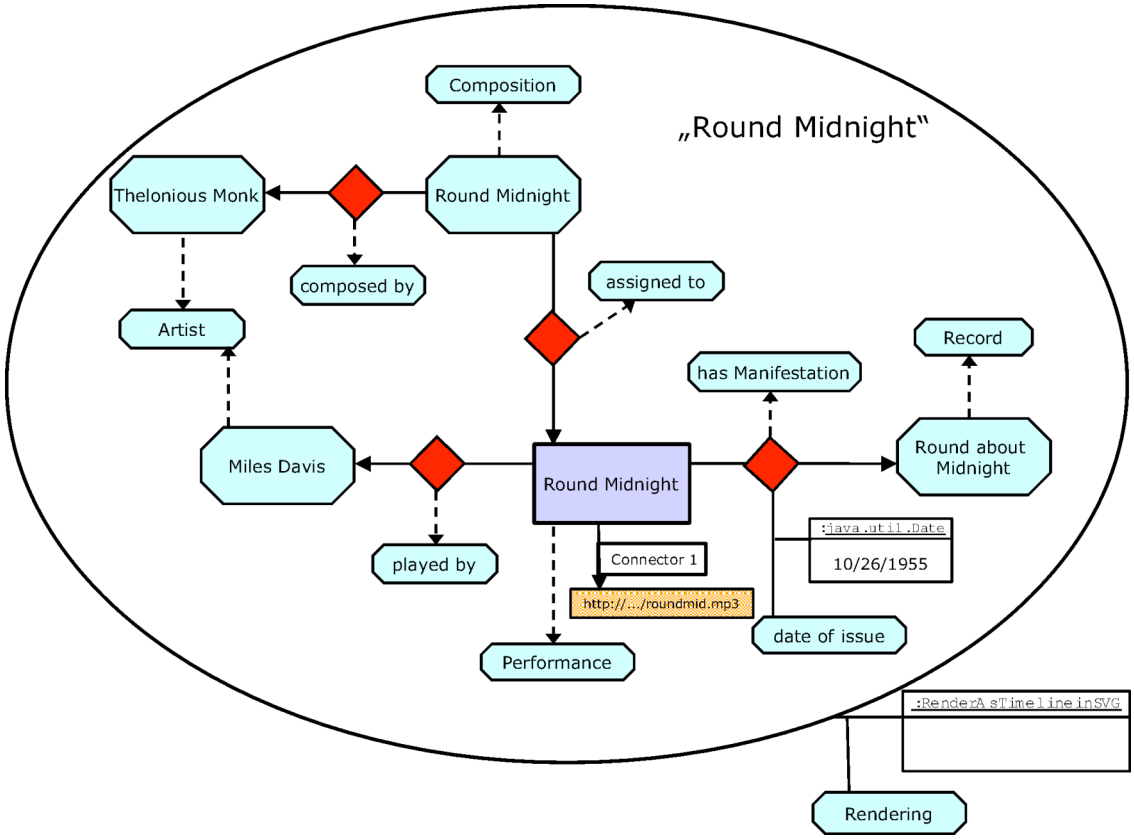

Fig. 23. Knowledge about the song "Round Midnight" represented by an Emmo

The functional aspect, finally, enables the Emmo to support different renditions of the knowledge it contains. To demonstrate this, we have realized an operation that, being passed a time interval as its parameter, produces an SVG timeline rendition (see screenshot of Figure 24) arranging important events like the foundation of bands, the birthdays and days of death of artists, etc. around a timeline. More detailed information for each event can be gained by clicking on the particular icons on the timeline.

Further operations could be imagined, e.g., operations that provide rights clearance functionality for the music files contained in the Emmo, which is a crucial issue in music sharing scenarios.

Our Emmo container infrastructure provides a capable storage foundation for semantic jukeboxes. Their ability to fine-grainedly manage Emmos as well as their scalability allowing

them to be deployed as both, as small-scale file-based and as large-scale database server configurations. Thus, Emmo containers constitute suitable hosts for knowledge-enriched music libraries of private users as well as libraries of professional institutions such as radio stations. Capable of exporting and importing Emmos to and from bundles, Emmo containers also facilitate the sharing of music between different jukeboxes. Their versioning support even allows it to move from mere content sharing scenarios to collaborative scenarios where different users cooperate to enrich and edit Emmos with their knowledge about music.

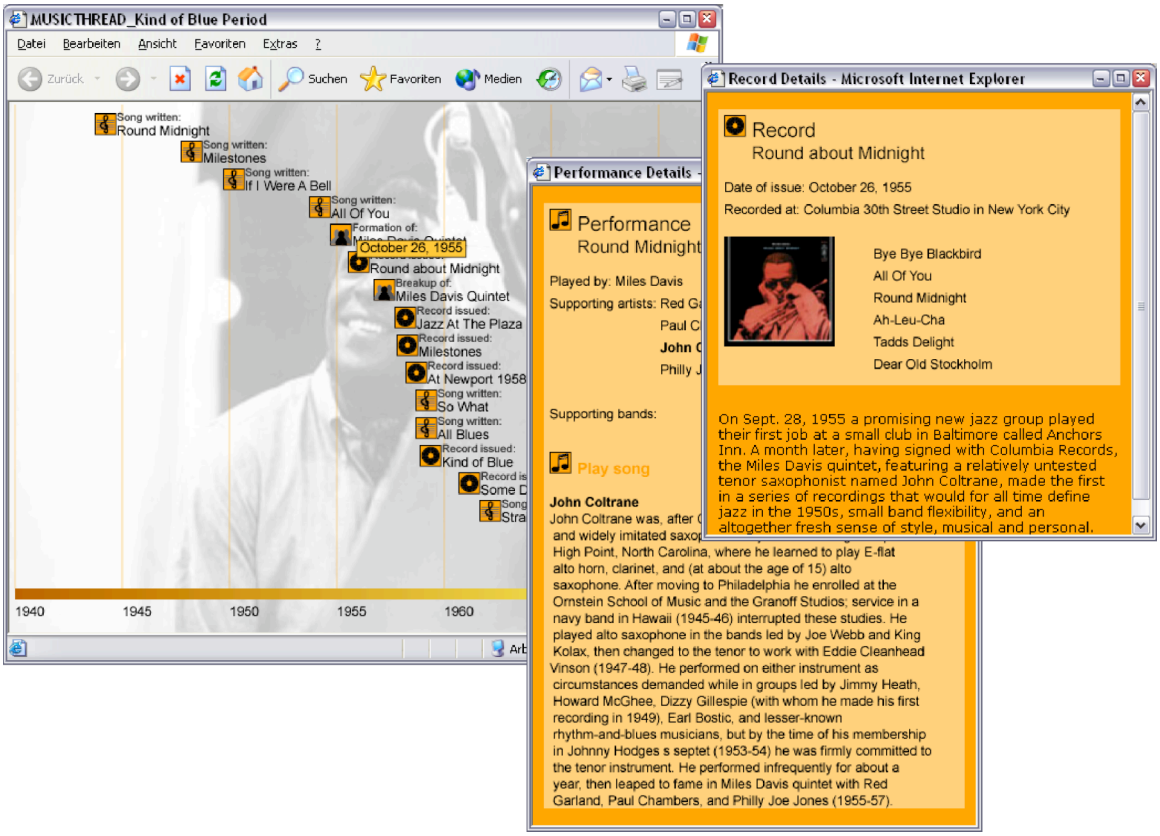

Fig. 24. Timeline rendition of a music Emmo

CONCLUSION

Current approaches to semantic multimedia content modeling typically regard the basic media which the content comprises, the description of these media, and the functionality on the content as conceptually separate entities. This leads to difficulties with multimedia content sharing and collaborative applications. In reply to these difficulties, we have proposed

Enhanced Multimedia Meta Objects (Emmos) as a novel approach to semantic multimedia content modeling. Emmos coalesce the media of which multimedia content consists, their semantic descriptions, as well as functionality on the content into single indivisible objects. Emmos in their entirety are serializable and versionable, making them a suitable foundation for multimedia content sharing and collaborative applications. We have outlined a distributed container infrastructure for the persistent storage and exchange of Emmos. We have illustrated how Emmos and the container infrastructure were successfully applied for the sharing and collaborative authoring of multimedia-enhanced intertextual threads in the CULTOS project and for the realization of a semantic jukebox.

We strive to extend the technological basis of Emmos. We are currently developing a query algebra, which permits declarative querying of all the aspects of multimedia content captured by Emmos, and integrating this algebra within our Emmo container implementation. Furthermore, we are wrapping the Emmo containers as services in a peer-to-peer network in order to provide seamless search for and exchange of Emmos in a distributed scenario. We also plan to develop a language for the definition of ontologies that is adequate for use with Emmos. Finally, we are exploring the handling of copyright and security within the Emmo model. This is certainly necessary as Emmos might not just contain copyrighted media material but also carry executable code with them.

REFERENCES

Ayars, J., Bulterman, D., Cohen, A., et al. (2001). Synchronized Multimedia Integration Language (SMIL 2.0). W3C Recommendation, World Wide Web Consortium (W3C).

Baumeister, S. (2002). Enterprise Media Beans TM Specification. Public Draft Version1.0, IBM Corporation.

Benari, M., Ben-Porat, Z., Behrendt, W., Reich, S., Schellner, K., & Stoye, S. (2002). Organizing the Knowledge of Arts and Experts for Hypermedia Presentation. Proceedings of the Conference of Electronic Imaging and the Visual Arts, Florence, Italy.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. Scientific American.

Boll, S., Klas, W., & Westermann, U. (2000). Multimedia Document Formats - Sealed Fate or Setting Out for New Shores? Multimedia - Tools and Applications, 11(3).

Brickley, D., & Guha, R.V. (2002). Resource Description Framework (RDF) Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft, World Wide Web Consortium (W3C).

Chang, H., Hou, T., Hsu, A., & Chang, S. (1995). Tele-Action Objects for an Active Multimedia System. Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS 1995), Ottawa, Canada.

Chang, S., & Znati, T. (2001). Adlet: An Active Document Abstraction for Multimedia Information Fusion. IEEE Transactions on Knowledge and Data Engineering, 13(1).

Daniel, R., Lagoze, D., & Payette, S. (1998). A Metadata Architecture for Digital Libraries. Proceedings of the Advances in Digital Libraries Conference, Santa Barbara, California.

Fensel, D. (2001). Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer, Heidelberg.

Ferraiolo, J., Jun, F., & Jackson, D. (2003). Scalable Vector Graphics (SVG) 1.1. W3C Recommendation, World Wide Web Consortium (W3C).

Gnutella. Retrieved 2003, from http://www.gnutella.com.

Grimson, J., Stephens, G., Jung, B., et al. (2001). Sharing Health-Care Records over the Internet. IEEE Internet Computing, 5(3).

ID3v2. Retrieved 2004, from http://www.id3.org.

ISO/IEC JTC 1/SC 29 (1997). Information Technology - Coding of Hypermedia Information - Part 5: Support for Base-Level Interactive Applications. ISO/IEC International Standard 13522-5:1997, International Organization for Standardization/International Electrotechnical Commission (ISO/IEC).

ISO/IEC JTC 1/SC 34/WG 3 (1997). Information Technology - Hypermedia/Time-based Structuring Language (HyTime). ISO/IEC International Standard 15938-5:2001, International Organization for Standardization/International Electrotechnical Commission (ISO/IEC).

ISO/IEC JTC 1/SC 34/WG 3 (2000). Information Technology - SGML Applications - Topic Maps. ISO/IEC International Standard 13250:2000, International Organization for Standardization/International Electrotechnical Commission (ISO/IEC).

ISO/JTC1/SC 32/WG 2 (2001). Conceptual Graphs. ISO/IEC International Standard, International Organization for Standardization/International Electrotechnical Commission (ISO/IEC).

ISO/IEC JTC 1/SC 29/WG 11 (2001). Information Technology - Multimedia Content Description Interface - Part 5: Multimedia Description Schemes. ISO/IEC Final Draft International Standard 15938-5:2001, International Organization for Standardization/International Electrotechnical Commission (ISO/IEC).

iTunes. Retrieved 2004, from http://www.apple.com.

Lagoze, C., Lynch, C., & Daniel, R. (1996). The Warwick Framework: A Container Architecture for Aggregating Sets of Metadata. Technical Report TR 96-1593, Cornell University, Ithaca, New York.

Lassila, O., & Swick, R.R. (1999). Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, World Wide Web Consortium (W3C).

Leach, P. J. (1998, February). UUIDs and GUIDs. Network Working Group Internet-Draft, The Internet Engineering Task Force (IETF).

Matena, V., & Hapner, M. (1998). Enterprise Java Beans TM. Specification Version 1.0, Sun Microsystems Inc.

Nejdl, W., Wolf, B., Qu, C., et al. (2002). EDUTELLA: a P2P Networking Infrastructure Based on RDF. Proceedings of the Eleventh International World Wide Web Conference (WWW 2002) Honolulu, Hawaii.

Newmann, D., Patterson, A., & Schmitz, P. (2002). XHTML+SMIL Profile. W3C Note, World Wide Web Consortium (W3C).

Pereira, F., & Ebrahimi T., (Eds.) (2002). The MPEG-4 Book. Pearson Education, California.

Reich, S., Behrendt, W., & Eichinger, C. (2000). Document Models for Navigating Digital Libraries. Proceedings of the Kyoto International Conference on Digital Libraries, Orlando, Kyoto, Japan.

Raggett, D., Le Hors, A., & Jacobs, I. (1999). HTML 4.01 Specification. W3C Recommendation, World Wide Web Consortium (W3C).