A formal framework for handling audiovisual materials

M. Baldoni*, C. Baroglio*, P. Bertolotti*, R. Del Pero†, A. Martelli*,
A. Messina†, G. M. Sacco*, and C. Schifanella*

*Dipartimento di Informatica — Università degli Studi di Torino
{baldoni,baroglio,bertolot,mrt,sacco,schi}@di.unito.it
†RAI Radiotelevisione Italiana - Centro Ricerche e Innovazione Tecnologica
{a.messina,r.del_pero}@rai.it

Abstract

In this paper we present a formal framework for the representation of audiovisual material produced or broadcasted by a TV company. Such a framework captures not only features related to the making of audiovisual objects but also features related to their inner structure. The framework exploits the compositional nature of audiovisual objects, and includes the possibility of dealing with broadcasts of audiovisual materials. A DAML+OIL ontology has been defined to represent the formal framework and has been used to implement a prototype system.

1 Introduction and motivations

The latest generation media channels, such as the Internet, see [14], and the advent of digital terrestrial TV, which is envisaged to represent the main context for the new services, require to the broadcasting industry a pervasive re-industrialization of the traditional production process, and in particular the definition of new *automated production models*. In this perspective, a primary objective is the development of advanced content management systems, that allow a wide range of operations to be tracked/performed automatically.

In the last years the international community of audiovisual content users and producers has seen the raise of many public and private metadata standards at different levels of complexity and with different scopes (e.g. [2], [3], [5], [10], [1]). By *metadata* it is commonplace, among the specialists of content management solutions, to mean *that information that relates to data*, where *data* is no more than the raw material, embodied in the traditional audiovisual asset formats (tapes) or in the computer file formats. The developed standards derive from a common approach to the problem, that can be briefly described as the identification of a catalogue of information elements and structures, that capture various aspects of the content description, from the low level video and audio description up to the semantic layer.

From the point of view of the content providers and publishers, however, these attempts are still insufficient because they do *not* yet supply a high level formalization of the *domain knowledge*, which would allow a more effective information interchange in real scenarios, where systems rarely use exactly the same data models, but rather they refer to inhomogeneous data schemes to implement similar or even the same domain conceptualization. A new approach would attempt at re-organizing and harmonizing the use of existing standards so to make an actual knowledge exchange possible during the whole content life-cycle. As a consequence, the rules for retrieving, representing and using knowledge should be reformulated, and these changes would reflect into innovative ideas for user applications.

As a first step in this direction, this work aims at defining a *formal framework* for the representation, processing, and the archiving of information with a *multimedia* nature. The definition of formal representation frameworks is particularly relevant also in the light of the emerging *Semantic Web* technologies [9], whose purpose is to extend the current Web with tools that enable resources to be semantically defined (by means of ontologies and ad hoc languages, such as DAML+OIL and, more recently, OWL [11]) and linked, so that they can be automatically retrieved, aggregated, and exchanged across different applications, by applying the reasoning techniques that have been (and are being) developed in the Semantic Web area

[13, 12]. The prototype that we present in this work, is based upon a sample ontology, written in DAML+OIL. Also in the field of *Multimedia Information Systems*, annotations and ontologies are used to improve the retrieval and to generate meaningful presentations, and some connections to the Semantic Web emerge.

In the seminal work by Bordegoni et al [15], a theoretical framework is proposed, whose task is to present information to the user in an effective way. The generation of presentations is based both on *presentation knowledge* and on *contextual knowledge*. More recent systems, based on [15], applying Semantic Web technology. In [16], the authors propose an ontology-driven transformation process to generate a coherent multimedia presentation which conveys the relations within the information to the user: the *semantic graph*, that represents the relations between the items in the domain, is converted in a data structure, called *structured progression*, used to create a multimedia presentation. These transformations are guided by different ontologies, that store the necessary knowledge. In [17], Little et al. generate presentations, based on media items annotated with the Dublin Core [2]: their system infers semantic relations between media objects, properly annotated and distributed across different archives; the obtained relations are mapped into spatial and temporal relationships between objects, which are, finally, expressed with SMIL[1] files, and so can be played as multimedia presentations.

In this work, we use Semantic Web technology in a different domain of application: the heterogeneity of audiovisual materials (AVM) in a television company needs a semi-automatic annotation system, in order to obtain a more efficient management of data, based on their semantic description. In particular, the motivations and requirement specifications for this work come directly from the experience of RAI, the largest Italian television company. Thus, we will focus on information about the AVM, that is produced or broadcasted by the company. The paper is organized as follows. Section 2 introduces in an intuitive way the main concepts at the basis of the formal framework, which in turn is reported in Section 3. In Section 4 the concepts introduced in the formal framework are represented in the DAML+OIL language, thus obtaining the AVM-DAML ontology. The ontology has been used to implement a prototype system for managing audiovisual objects, that is accessible through a web interface. Conclusions follow.

[1] A W3C standard markup language.

2 Adding semantics to AVM representations

TV companies own huge archives of AVMs, that can be broadcasted, reprocessed, bought, sold, or used to produce new AVMs. Each AVM is characterized by many *properties*, some of which are legal grants and credits. One of the main problems, that TV companies must face, is the *management* of such properties. Indeed, in most cases, the process of annotating television products is simply a manual process. There are people from the staff whose role is specifically that of watching television programs, listing, and storing in an ad hoc database, the properties of the programs. An *automatic annotation system*, able to associate properties with products on the basis of some formal requirement, would be highly desirable, but not straightforward.

The difficulties are mainly due to the different nature of these properties, and their different granularity levels. Some properties specifically concern the format level of the AVM (we call this level of representation *encoding*). This is the case of the distinction between an mpeg2 or a divx video. Some other properties might still concern the overall AVM, while being unrelated from the representation format (they concern the *abstract concept*). One example is the distinction between a black-and-white and a color movie. Moreover, when an AVM is the result of the *composition* of other AVMs, it might inherit some properties from its components. For instance, if a program contains a soundtrack, with a copyright, the latter must appear in the list of the grants of the final program.

To make this discussion more concrete, we will consider, as a reference example along the paper, the movie *Metropolis*. The director of the movie is Fritz Lang: this property is independent of the specific copy of Metropolis that is in the archives and of its format (abstract concept property). However, depending on the physical support, on which the movie is recorded, a different reading device is to be used (encoding property). Moreover, if a colored version of the movie is produced, the credits about by whom, how, when, and where this was done will be associated to the colored version. Notice, that the properties of the new version include many properties of the original version; part of them, actually, does not depend on the version (e.g. the actors).

As an immediate consequence of these observations, in our framework we represent AVMs by means of a two-level description, where we separate the level of *encodings* from the level of their semantic interpretations, where the term interpretation is related to the conceptual content of such objects, the *abstract concept*.

In particular, different encodings may have the same interpretation, and the interpretation does not necessarily change when the content is changed, as it may happen in the case of the colored version of Metropolis. Then, an encoding must not necessarily correspond to any *final* product. In this case it plays the role of a *partial production*, with its associated properties, that will be properly propagated to the new productions realized out of it. Another important aspect that we model regards the *broadcast* of AVMs. Broadcasts, as well as encodings and interpretations, have properties that the company needs to manage. For the sake of complexity, broadcasts can be recorded producing new encodings.

The resulting framework, which not only models the distinction between different semantic characterization levels, but also the composition aspect of TV programs, is what we need to develop a *semi-automatic* annotation tool. We stress the semi-automatic nature of such a tool, since we are not suggesting a way to provide annotations from scratch for a multimedia product, but rather, we are providing a framework that allows to automatically *reuse*, and possibly manually *extend with new information*, the annotations of those AVMs that play the role of data in the production of a new AVM.

3 The semantic framework

In this section, we will introduce the basic notions related to our proposal, which are: *encodings, abstract concepts, interpretations, and events.*

Assume two friends are talking about *Metropolis*. They both probably refer to Fritz Lang's movie, produced in 1926, and this is true independently of where and when they watched that movie. When we say "watching Metropolis", we refer to (i) an *abstract concept* (Fritz Lang's movie), and (ii) a corresponding *encoding* (e.g. a VHS tape), associated with the abstract object. The duality is captured by the notion of *interpretation*, that associates abstract concepts with encodings. The same abstract concept can be associated with several encodings, which may be seen as physical manifestations of the concept. Back to the example, the DVD encoding of Metropolis is different from the taped one, but they are both associated to the same concept, the movie Metropolis. Observe that the interpretation that relates encodings and concepts is *not unique*. For instance, returning to the colored version of Metropolis, a film reviewer will probably consider it something different from the original B/W, while a less expert viewer would associate both versions to the same movie.

3.1 Encodings

Definition 3.1 (ENCODING) *An encoding E is a 5-tuple $\langle id, hs, tp, al, sl\rangle$, where: id is a unique identifier, hs captures the history of the encoding (we will come back to this notion soon), tp is the type, al is an attribute list, and sl is a (possibly empty) support list. An attribute is a pair $\langle$attribute-name, attribute-value$\rangle$.*

In the following, a set of encodings E will be denoted by $\mathbf{E}$, while $\mathbf{A}$ will denote the set of all possible encoding attributes, and $\mathbf{S}$ the set of all physical supports. Clearly, *al* and *sl* are subsets of $\mathbf{A}$ and $\mathbf{S}$, respectively.

Every encoding also has a *type*, that gives the information about what kind of encoding algorithm has been used, and thus, implicitly, provides a descriptions of the representation that has been used. As an example, consider different image files. The same image can have different formats, such as jpeg, bitmap, or png. To display the image, the format of the file will have to be properly interpreted, and transformed in a matrix of colored pixels.

The set of all the encoding types will be denoted by $\mathbf{T}$, and $\mathbf{E}_{tp}$ will denote the subset of encoding E in $\mathbf{E}$ having type *tp*. Thus, the set $\mathbf{E}$ of the encodings is equal to $\bigcup_{type\in\mathbf{T}} \mathbf{E}_{tp}$.

Functions mapping encodings into encodings (that is, operations having encodings as their arguments) may be defined. The encodings in both the domain and the co-domain are usually typed. An example of such an operation is the transformation of a GIF encoding into a corresponding (i.e., representing the same image) PNG encoding.

Definition 3.2 *An operation on encodings is a function $op : \boldsymbol{E}_{tp1} \times \boldsymbol{E}_{tp2} \times \ldots \times \boldsymbol{E}_{tpk} \rightarrow \boldsymbol{E}_{tp}$.*

There is no a priori hypothesis on the types of encodings involved in an operation, nor on the type of the result. In particular, there is no requirement that those operations, that combine homogeneous encodings (that is, encodings of the same type), return an encoding that is homogeneous to the input ones. This choice allows us to have a very general and flexible model, and lets us characterize several transformation functions, such as Composition, Association, Decomposition, Dis-Association. For example, it is possible to combine a sound track with a video, and obtain an audiovisual product. Instead, if we combined a set of jpeg images, we could obtain a new jpeg image, as well as an image in another format, or maybe an animation. Of course, every single operation has its own restrictions/applicability requirements, that we see as part of its specification.

It may be worth noticing also that the types of the involved encodings can be any, no matter whether they are elementary types (e.g. jpeg, or mp3, or divx), or compound types (e.g., $\langle divx, mp3 \rangle$).

We are specifically interested in operations that *keep track of the history*, and allow to know how any given encoding was produced. We want indeed to be able to distinguish between primitive encodings, and encodings that result from operations applied to some arguments (that, in turn, might be primitive encodings, or results).

Suppose we have an operation $op : \mathbf{E}_{tp1} \times \mathbf{E}_{tp2} \rightarrow \mathbf{E}_{tp1}$ combines pairs of encodings like $\langle id_i, hs_i, tp_i, al_i, sl_i \rangle$ and $\langle id_j, hs_j, tp_j, al_j, sl_j \rangle$, and returns encodings such as $\langle id_z, hs_z, tp, al_z, \emptyset \rangle$, where id_z is a new identifier, hs_z is the 3-tuple $\langle op, id_i, id_j \rangle$, tp is the type of the range of op and al_z is an attribute list.

Another important information maintained in the model is the *storage list*, where the different (possibly none) physical storages, on which the encodings are recorded, are listed. Multiple copies for the same encoding may exist, and there are no *a priori* restrictions on the supporting physical medium (the only restrictions concern the feasibility, from the physical point of view, of the recording operation: for example, it would be impossible to record an analogue audio on a CD).

This information will be useful, for example, when we will need to produce new material out of products stored in the archive, and the data in the model will be used to find the specific physical storage media (disks, tapes, cassettes) that have to be reproduced, for the new production.

To model the recording of a product on a physical support, the only relevant information is the identifier of the support. The recording operation can thus be formalized as follows.

Definition 3.3 *A recording operation is a function* $rec : \boldsymbol{E}_{tp} \times \boldsymbol{S} \rightarrow \boldsymbol{E}_{tp}$.

Given an encoding $\langle id, hs, tp, al, sl \rangle$ and a support identifier s, the new encoding $\langle id, hs, tp, al, \{s\} \cup sl \rangle$ is returned (the returned encoding differs from the given one only because of the support list, that is enriched with the new support identifier).

The only element in the definition of encoding that still needs some comment is the *attribute list*. It is a list of attributes, that keeps track of some properties characterizing the encoding, like the author, the production date, the duration ... They are mostly metadata about the encoding. Some of these properties are known from the very beginning. This is the case, for example, for the creation date. Some others are captured and associated to the encoding at a certain point of its life. This is the case, for example, for grants, copyrights, ...

To allow the model to expand the attribute list at any time, a characterization operation is also defined.

Definition 3.4 *A characterization operation is a function* $cr : \boldsymbol{E}_{tp} \times \boldsymbol{A} \rightarrow \boldsymbol{E}_{tp}$.

Given an encoding $\langle id, hs, tp, al, sl \rangle$ and an attribute a the new encoding $\langle id, hs, tp, \{a\} \cup al, sl \rangle$ is returned.

Thus, to give a meaning to the newly produced encoding, a new interpretation step will be needed. This is what lets us capture the case discussed in the introductory example. Assume we are given an encoding of the movie Metropolis, and a transformation function that, applied to it, returns a new encoding, which is the result of cutting out the first ten minutes. Depending on the interpretation function, the two encodings will be assigned the same meaning, or different meanings. Of course, different interpretation functions might exist, thus matching different users attitudes. In general, we have different kinds of interpretations, that choose to unify or keep distinct objects interpretations similar (but not identical) to each other, according to the context, and the role they will play. For example, sometime the movies broadcasted on television are slightly different from the original ones. The transformation is often due to television timing requirements, and to interruptions for advertisements, that force the projection to fit in a given scheduled time slot, which doesn't necessarily match the exact duration of the original product. But, even if the transmitted movie is not the original one, and an expert user could tell it, the same title is kept, and the transmission is associated to the same concept as the original encoding.

3.2 Abstract Concepts and Interpretations

At the encoding level, anyway, the difference between the two products is coded. It is highly possible that in different contexts, for example, for historical/cinematographic studies, they are given different interpretations.

Definition 3.5 (ABSTRACT CONCEPT) *An abstract concept I is a tuple* $\langle id, al \rangle$ *, where id is a unique identifier and al is an attribute list.*

Definition 3.6 (INTERPRETATION) *An interpretation for a set of encodings* $\boldsymbol{E}$ *is a tuple* $\langle \boldsymbol{I}, i \rangle$*, where* $\boldsymbol{I}$ *is an abstract concept domain and i is a partial function, named interpretation function, from* $\boldsymbol{E}$ *to* $\boldsymbol{I}$.

Intuitively, not every encoding is associated to an abstract concept (this is why interpretation function is

a partial function). For example, some encoding can represent partial data, to be used in the production of a (meaningful) program. Moreover, different encodings may be associated with the same concept: the interpretation function is not required to be injective.

Given any interpretation function i, it is important to be able to model the reverse behavior, to go back, from any given abstract concept, to the set of encodings that i associates to it. We denote with $\bar{i}$ the function that models this behavior. Notice that $\bar{i}$ is not simply the reverse function i^{-1} of $i : \mathbf{E} \rightarrow \mathbf{I}$, since the range of $\bar{i}$ is $2^{\mathbf{E}}$.

Definition 3.7 *Let $\langle \boldsymbol{I}, i \rangle$ be an interpretation for a set of encodings $\boldsymbol{E}$. Given an encoding $E \in \boldsymbol{E}$ and an abstract concept $I \in \boldsymbol{I}$ we denote with $\langle \boldsymbol{I}, i[E \rightarrow E'] \rangle$ a new interpretation where $i[E \rightarrow E'](E'') = i(E'')$ for any $E'' \in \boldsymbol{E}$ such that $E'' \neq E$ and $i[E \rightarrow E'](E) = E'$.*

Definition 3.8 *Any operation $op : \boldsymbol{E}_{tp1} \rightarrow \boldsymbol{E}_{tp2}$ is said to be invariant w.r.t. an interpretation $\langle \boldsymbol{I}, i \rangle$, if for every encoding $E \in \boldsymbol{E}_{tp1}$, the equality $i(E) = i(op(E))$ holds.*

3.3 Events

Another entity that TV companies need to manage is AVM broadcast. Such entities have own properties (such as time and channel, sponsors, transmission rights), which are not related to the AVM properties seen so far. Observe that the same encoding of a concept can be played several times. Basically, different play-outs of any encoding will differ on the values of their properties. In our framework we will call *event* any broadcast.

Definition 3.9 (EVENT) *An event EV is a 4-tuple $\langle id, E, al, t \rangle$ where id is a unique identifier, E is an encoding, al is a set of attributes, and t is the time instant in which the encoding E has been displayed.*

The set of attributes al describes properties specific to the event. Given a set of encodings $\mathbf{E}$, we denote by $\mathbf{EV}_E$ a set of events based on $\mathbf{E}$.

3.4 Concepts

Another aspect that we want to consider, is the conversion, in a machine readable format, of the mental process that permits to define and to create a television product. These processes are variable, generically long and complex, and they include different steps of development: from the drafting of a script to the construction of the scenes, to the casting and so on. At the actual state of the project, we have not a complete formalization of the process, but we believe that the idea is promising.

Definition 3.10 (CONCEPT) *A concept Co is a pair $\langle id, al \rangle$ where id is a unique identifier and al is a set of attributes.*

Co denotes the set of all concepts. In general, every encoding can be associated with a concept, that describes the process after which the encoding is created. Observe that a similar idea should be valid for concepts, because they are commonly the result of a sequence of production steps.

Definition 3.11 (ENCODING-CONCEPT ASSOCIATION) *An encoding-concept association is a partial function $g : \boldsymbol{EV}_C \rightarrow \boldsymbol{Co}$*

Moreover, framework should includes an operation to characterize concepts, in order to deal with property adjunction.

Definition 3.12 (CONCEPT CHARACTERIZATION) *The characterization of a concept Co is a function $cr : \boldsymbol{Co} \times \boldsymbol{A} \rightarrow \boldsymbol{Co}$.*

Given a concept $\langle id, al \rangle$ and an attribute a, cr returns the new concept $\langle id, \{a\} \cup al \rangle$.

4 The prototype

In order to test the proposed framework, we have developed a prototype application software, that we briefly describe in this section. The first step has been to write in a machine-interpretable form the concepts that make the formal framework. To this aim we have chosen the DAML+OIL (now OWL) language. This choice is motivated by the particular context in which this project is located, characterized by a strongly distributed environment and a high heterogeneity of tools and of knowledge bases used within it. The DAML+OIL language, the standard to represent the semantic content of documents accessible via web, is based on the definition of classes and the inheritance of properties in the class hierarchy and permits mechanisms of inference. It is a machine-oriented language, used to exchange and to process information. This choice is also motivated by the presence of different tools for using this language: some editors ([7], [6], [8]) and some API to develop programs (for example, the HP suite Jena [4] is used to keep DAML+OIL ontologies in Java).

We have, so, obtained the ontology graphically described by Figures 2 and 3, whose root element is class

Entity. This class is not part of the formal framework and has been introduced for implementative issues. The classes that are direct successors of Entity correspond, indeed, to the concepts that make the formal framework (encoding, abstract concept, and so forth). By means of the classes at this level of representation, it is possible to capture the overall structure of an AVM, its history, its properties, references to other AVMs that have been used to compose it, generally speaking all the kinds of information that are necessary for managing AVMs. However, in order to give more precise descriptions of AVMs it might be useful in some implementations to specialize the ontology adding some other classes, such as those reported in Figure 4 at the lowest level (e.g. Film or TV Program). The reported classes are to be considered just as an example, and not as an exhaustive representation.

The prototype consists of:

1. the AVM-DAML ontology, described above;
2. the JAVM package, a Java package, that uses Jena 1.6, which supplies all the methods for handling the AVM-DAML ontology;
3. a web interface, written in JSP, which uses JAVM and that has been developed in order to test the framework the afore mentioned components.

4.1 The AVM-DAML ontology

As already seen, an ontology is a shared document, that contains the formal description of the concepts of a given domain, identifying the main classes, their organization, their properties and the most meaningful relations between them. Our domain includes:

1. encodings (as described in Section 3);
2. abstract concepts (as described in Section 3): they can be split in schedule elements, i.e., the single units in which an abstract concept can be divided, and composite concept, i.e., a more complex element, like film, TV program and so on;
3. events (as described in Section 3);
4. interpretations (as described in Section 3);
5. physical support: the storage support for the encodings;
6. operations: the permitted operations on encodings, i.e., extraction, union and registration;
7. people: those who manage the domain.

PROPERTY	RANGE	CHARACT.
channel	string	Datatype, unique value
moment	dataTime	Datatype, unique value
transmittedEncoding	*Encoding*	Object, unique value

Table 1. Properties of the class *Event*.

Our ontology AVM-DAML implements these elements, as described in Figure 2, 3, 4 and 5. We have the following classes: *Encoding*, *AbstractConcept*, *ScheduleElement*, *CompositeConcept*, *Film*, *Telefilm*, *TVProgram*, *Event*, *Interpretation*, *PhysicalSupport*, *Operation*, *Extraction*, *Union*, *Registration* and *Person*. Moreover, we have added a most generic class *Entity*: its properties are common to all elements in the ontology.[2]

In DAML+OIL we list, for each class, the set of its properties and, for each property, we indicate its name, its domain (i.e., the range of values that it can assume), the type of the range (i.e., if it is a *DataType* defined in DAML+OIL or if it is an object defined in the ontology itself) and, finally, if it is a key for the class (i.e., if the property has to assume a unique value). As an example, let us consider the class *Event*, that represents the AVM broadcast: their properties are the TV channel, a *string* with unique value, the time, a *dataTime* with unique value, and the transmitted encoding, an object with unique value (see Table 1 and Figure 1 for the corresponding code).

4.2 The JAVM package for the management of the AVM-DAML ontology

The JAVM package, implemented in Java, offers an interface for the management of the AVM-DAML ontology, introduced in the previous section. Current version of JAVM uses the package Jena, version 1.6, distribuited by Hewlett-Packard [4] (see Figure 6). For every class introduced in AVM-DAML, we have defined an analogous class in JAVM, keeping the hierarchical structure. The skeleton of the JAVM classes is obtained automatically from the AVM-DAML ontology. This feature is extremely useful since the ontology can be modified or updated in order to consider new aspects, and in this case corresponding reorganization in the package should occur. In this way, the task of the programmer is to define only the methods that represent the business logic of the instance.

[2]The AVM-DAML ontology is available at `http://www.di.unito.it/~alice/AVM-DAML/AVM-DAML-V4.3.daml`.

```
<daml:Class rdf:ID="Event">
    <rdfs:subClassOf
    rdf:resource="#Entity"/>
</daml:Class>

<daml:DatatypeProperty
   rdf:ID='channel'
   rdf:type='UniqueProperty'>
   <daml:domain
      rdf:resource='#Event'/>
   <daml:range
      rdf:resource='string'/>
</daml:DatatypeProperty>

<daml:DatatypeProperty
   rdf:ID='moment'
   rdf:type='UniqueProperty'>
   <daml:domain
      rdf:resource='#Event'/>
   <daml:range
      rdf:resource='dataTime'/>
</daml:DatatypeProperty>

<daml:ObjectProperty
   rdf:ID='transmittedEncoding'>
   <daml:domain
     rdf:resource='#Event'/>
   <daml:range
     rdf:resource='#Encoding'/>
</daml:ObjectProperty>
```

Figure 1. Excerpt of the AVM-DAML ontology.

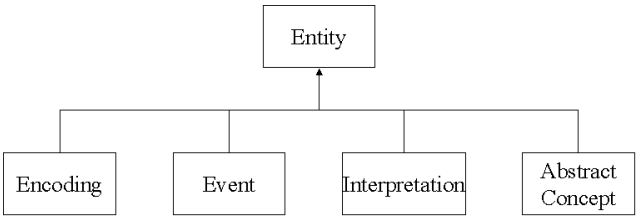

Figure 2. Main elements of the model defined in the ontology.

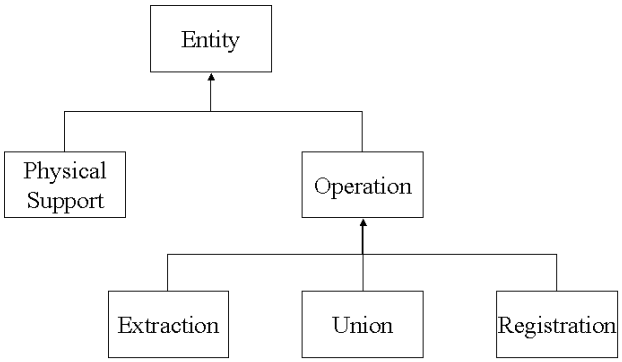

Figure 3. Classes that define the operations.

Every AVM is represented by an instance of the AVM-DAML ontology, and can be handled by means of the methods of the corresponding JAVM class. From an implementation point of view, this is obtained as described by Figure 7: every instance of a class in JAVM has a reference to an instance of the corresponding class in AVM-DAML, that represents its state and that is can be handled by means of the methods in the Jena package. Every JAVM class, actually, defines the *business logic* of its instances, e.g., how to carry out the union operation between two encodings.

By using Jena, version 1.6, the persistency is obtained by dumping the states into ascii files. During the execution the whole XML tree is kept in memory. The latest version, instead, allows the use of databases not only for dumping the states but also during execution.

In JAVM, the class *OntologyManagerDAML* contains the implementation of the interface to the AVM-DAML ontology through the package Jena. Then, this class contains the methods to download/save the model AVM-DAML, to create/remove and manage the properties of the instance AVM-DAML; with this class, the use of Jena and its possible upgrades are hidden to the programmer.

We have also deployed a web application that permits, thorough a web browser, to use the methods ex-

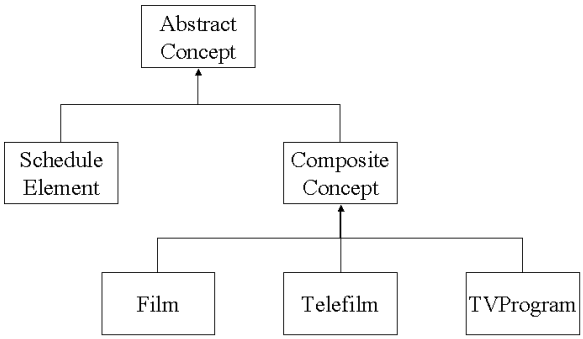

Figure 4. Added classes.

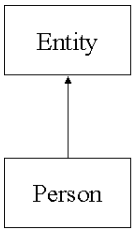

Figure 5. The class Person.

posed by the package JAVM.

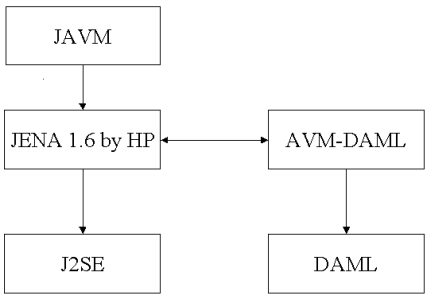

Figure 6. Relation between JAVM and Jena.

5 Conclusions and Future Work

In this work, we have proposed a *formal framework* for the representation, processing, and the archiving of audiovisual materials (AVM), that is produced or broadcasted by RAI, the largest Italian television company. We have identified two key concepts, the encoding and the abstract concept, which are related by interpretation functions. Notice that many such functions may co-exist, each of them capturing a specific point of view of a person, a class of people, a company, thus allowing a high degree of personalization. We have

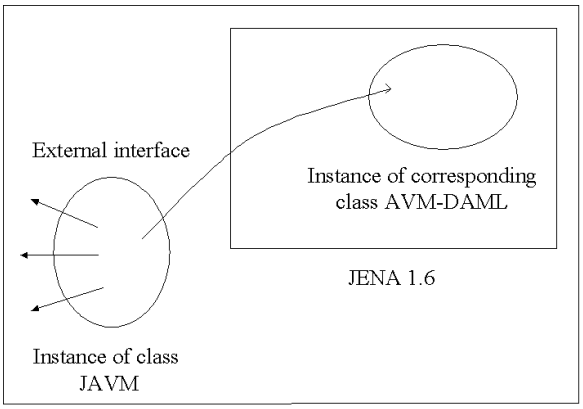

Figure 7. Instance JAVM vs instance AVM-DAML.

also presented a *prototype system* that aims at testing the proposed model. This system includes an ontology (AVM-DAML), written in the language DAML+OIL, and a set of Java classes to handle such an ontology. The ontology includes the main concepts of the model and their relations, and some elements with an applicative purpose (such as *film* and *director*).

Our approach is open to future development, that can either be technological (e.g. passing to version 2.0 of Jena and from DAML+OIL to the OWL language [11]) or address issues related to the presentation and the use of knowledge.

Concerning the use of the knowledge represented by means of the ontology, it would be interesting to test some techniques of *information retrieval* based on reasoning, that, by exploiting inference mechanisms, permit to deduce implicit relations between the elements. The toolkit Jena already supplies a few *reasoners*, nevertheless, we believe that the integration with external reasoning mechanisms should not be neglected.

On the side of the application domains, we think that, although, some of the choices that we have taken surely depend on the specific characteristics of the application domain, other application domains could benefit of the approach that we have proposed: all those domains in which it is necessary to store data subject to versioning, and where it is necessary to associate a same semantics to different objects. As an example, consider the problem of handling different editions of a novel, some of which contain only the text, while others contain also comments by experts in the field, and others are oriented to high-school students.

Finally, the efficiency can be augmented, using new mechanisms of persistence: they allow to maintain the ontology in a relational database.

References

[1] BBC (British Broadcast Corporation) SMEF (Standard Media Exchange Framework). http://www.bbc.co.uk/guidelines/smef.

[2] Dublin Core Metadata Initiative. http://dublincore.org.

[3] EBU/UER (European Broadcasting Union/Union Europienne de Radio-Télévision) Metadata Exchange Standards. ttp://www.ebu.ch/pmc_meta.html.

[4] The jena semantic web toolkit - version 1. http://www.hpl.hp.com/semweb/jena1.htm.

[5] Multimedia content description framework, ISO/IEC JTC1/SC29/WG11 standard. http://mpeg.telecomitalialab.com/standards/mpeg-7/mpeg-7.htm.

[6] Oiled, ontology editor. http://oiled.man.ac.uk/.

[7] OntoEdit, ontology editor. http://www.ontoprise.de/.

[8] The protégé project. http://protege.stanford.edu/.

[9] Semantic Web Organization. http://www.semanticweb.org.

[10] SMPTE (Society of Motion Picture and Television Engineers) Metadata Dictionary, SMPTE Recommended Practice RP 210.2.

[11] Web-ontology working group. http://www.w3.org/2001/sw/WebOnt/.

[12] S. Abdennadher, J. Alves Alferes, G. Antoniou, U. Assmann, R. Backofen, C. Baroglio, P. A. Bonatti, F. Bry, W. Drabent, N. Eisinger, N. E. Fuchs, T. Geisler, N. Henze, J. Maluszynski, M. Marchiori, A. Martelli, S. Carro Martinez, H. Jurgen Ohlbach, S. Schaffert, M. Schroeder, K. U. Schulz, U. Schwertel, and G. Wagner. Automated reasoning on the web. *Communications of Applied Logic*, 2004. To appear.

[13] M. Baldoni, C. Baroglio, L. Giordano, A. Martelli, and V. Patti. Reasoning about communicating agents in the semantic web. In F. Bry, N. Henze, and J. Maluszynski, editors, *Proc. of the 1st Int. Workshop on Principle and Practice of Semantic Web Reasoning, PPSWR 2003*, volume 2901 of *LNCS*, pages 84–98, Mumbai, India, 2003. Springer.

[14] L. Boch, R. Del Pero, G. Dimino, and A. Messina. Automatic Management of A/V Content Production for New Media Business. In *IBC 2001 Conference Publication*, pages 95–106, 2001.

[15] M. Bordegoni, G. Faconti, S. Feiner, M.T. Maybury, Th. Rist, S. Ruggieri, P. Trahanias, and M. Wilson. A Standard Reference Model for Intelligent Multimedia Presentation Systems. *Computer Standards and Interfaces*, 18(6-7):477–496, December 1997.

[16] J. Geurts, S. Bocconi, J. van Ossenbruggen, and L. Hardman. Towards Ontology-driven Discourse: From Semantic Graphs to Multimedia Presentations. In *Proc. of Second International Semantic Web Conference (ISWC2003)*, pages 597–612, Sanibel Island, Florida, USA, October 20-23 2003.

[17] S. Little, J. Geurts, and J. Hunter. Dynamic Generation of Intelligent Multimedia Presentations through Semantic Inferencing. In *6th European Conference on Research and Advanced Technology for Digital Libraries*, pages 158–189, Springer, September 2002.