Integration of Semantic Web Technology in an Annotation-based Hypervideo System

Olivier Aubert, Pierre-Antoine Champin, and Yannick Prié

LIRIS, University Claude Bernard Lyon 1
F-69622 Villeurbanne Cedex, France
firstname.lastname@liris.cnrs.fr
http://liris.cnrs.fr/firstname.lastname

Abstract. This article discusses the integration of semantic web technologies (ontology and inference) into audiovisual annotation based models and systems. The Advene project, aimed at all purpose hypervideo generation from annotated audiovisual documents, is used as a testbed. Advene principles and the Advene prototype are first presented, before a discussion on how ontology and reasoning can easily be integrated into the Advene framework. Some motivating examples are proposed, and our proposals and related works are discussed.

1 Introduction

This article has two primary goals and one angle of vision. The first goal is to present the Advene (Annotate Digital Video, Exchange on the NEt) project, rationale and prototype, as a powerful set of ideas and tools for designing and testing innovative uses of video through rich video annotation and hypervideos generation. The second goal is to present how semantic web technologies can be used in the Advene framework, in order to provide extra features mainly related to ontological reasoning. The angle of vision we adopt throughout the article is more related to the audiovisual annotation and hypermedia document engineering than to the semantic webour main concern is to study what semantic web technologies can bring to audiovisual annotation, hypervideo construction and audiovisual information systems.

We generally consider [1] that full audiovisual information systems (AVIS) should provide users with the possibility to search into video bases using indexes, to select from the retrieved documents the most relevant for the current task, and to reuse parts of documents, mainly in new hypermedia documents (simply watching them being a very simple reuse for instance). We also claim that annotations are the pivot to such audiovisual information systems, as they provide all the necessary information to search, select and manipulate audiovisual documents and fragments.

As a means to understand present annotation-based audiovisual information systems and to design future ones, we choose to use the hypervideo concept. *Hypervideos* [2] refer to hypermedia documents that are constructed from *annotated audiovisual documents* information: digital audiovisual documents (moving images and related sounds) and annotations for these audiovisual documents, which are digital pieces of information in relation with spatio-temporal fragments of the documents. Examples of hypervideos include web pages that refer to a video using some excerpts as key images,

video streams enriched with textual information and hyperlinks, reconstructed audiovisual streams, etc.

The next section of this article presents the Advene model and prototype for hypervideo engineering. The following section describes how we easily added semantic web reasoning capabilities to the Advene prototype thanks to its flexible model. The last section deals with some examples of semantic web-enriched uses of annotations in the context of Advene and more generally in the AVIS/hypervideo context.

2 Advene model and prototype

The Advene project aims at providing tools to exchange various analyses about movies stored in digital form (digital video files, DVDs...), and more importantly, offer the possibility to enhance and customize these analyses. Analyses are built upon annotations, which represent pieces of data of any type that are (spatio)temporally linked to the movie. The Advene prototype thus provides means to create and modify annotations, as well as to specify how they should be rendered in meaningful ways. Instead of exchanging the sole final form of an analysis, the Advene project makes it possible to rather exchange annotations and the specification of their visualisation, thus allowing end-users to customize data and visualisations in order to fit their needs.

We will see in this section how data is organized by the Advene model, and how the model is implemented in the current prototype[1].

2.1 The Advene model

It is commonly agreed that the handling of audiovisual contents has to use metadata, the audiovisual data itself being not fitted to indexing or querying without any preprocessing. Of the various existing approaches, let us give an overview of two important standards – MPEG-7[3] and Annodex[4] – and see how our proposal relates to them.

MPEG-7 aims at being the standard representation format to exchange metadata associated to audiovisual streams. It provides means to link metadata to portions of audiovisual documents. The MPEG-7 standard defines standard metadata, mostly focusing on low-level descriptors automatically extractable from the audiovisual document (colors, textures, shapes, audio characteristics...), as well as a way to specify additional metadata through XML Schema. It is used by some vendors, but a common complaint is the complexity of its model, which makes it difficult to use for simple things or for interoperation with other standards [5].

The Annodex[4] projects aims at creating a *continuous media web* where metadata is embedded in audiovisual documents, making them indexable and searchable. Aiming at simplicity, its *Continuous Media Markup Language* (CMML) is inspired by HTML, and allows to quickly edit metadata. After edition, CMML files are combined with the audiovisual documents. Annodex solves the simplicity issue (using an HTML inspired syntax), at the expense of a lack of structure. Moreover, it merges metadata with the audiovisual document, making it harder to use different metadata for the same audiovisual document.

[1] available from `http://liris.cnrs.fr/advene/`

The Advene model somehow aims at bridging the gap between both approaches: it provides a way to link metadata to audiovisual documents. It does not impose any constraints on the nature of metadata, and keeps metadata separate from the audiovisual document, so that they can evolve and be exchanged independently from each other.

Annotation structure We developed the Advene model based on our reflexion about hypervideos [1]. An *Annotated Audiovisual Document* (AAD) is an audiovisual document augmented with metadata. Processing both the audiovisual document and its accompanying metadata gives *views* on the AAD, some of them qualifying as *hypervideos*: views of the AAD that on the one hand use information from both the audiovisual document *and* the annotation structure, and on the other hand give access to the temporality of the audiovisual document.

In the Advene model, described more precisely in [2], the annotation structure consists mainly of *annotations*, that contain data and are linked through a temporal (possibly spatio-temporal) fragment to a specific portion of the audiovisual document. The structure of data contained in the annotations is not specified by the model: it can be any type of data (simple text, structured information, audio documents, office documents...).

In order to be usable, while retaining their genericity, annotations are flexibly structured: *annotation types* define the kind of content (through a MIME-type specification) held by annotations. Multiple annotation types can be used to describe a number of analysis facets. Moreover, *relations* allow to link annotations with each other, and are specified by *relation types*. Relation types define the types of annotations that can be linked, as well as an optional content MIME type for relations.

As annotation types and relation types define a certain point of view in the document analysis, they are grouped as meaningful sets called *schemas*. An Advene schema thus defines annotation types and relation types that form together an analysis framework.

Let us illustrate this structure through a simple example. Consider a movie containing a lot of flashbacks. The analysis of the temporal relations of the various narrative sequences (also called *diegetic chronology*) can be used to discourse about the narrative structure. We define an Advene schema called *diegetic chronology*, that contains two annotation types: *shot*, that represents a shot as the basic unit in movies and *diegetic sequence* that represents a chronologically consistent unit. A relation type, *followed by*, will allow to link a sequence to the following one in the diegetic chronology. Another schema, called *movie*, contains among other types a *character* annotation type, that represents a character. Figure 1 sums up these schema, figure 2 present how it is possible to annotate a movie using the diegetic chronology schema.

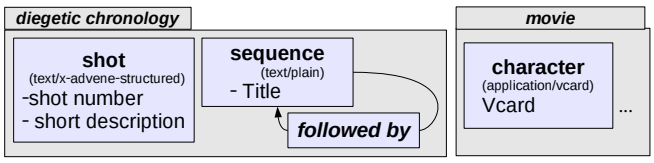

Fig. 1. The *diegetic chronology* and *movie* schema

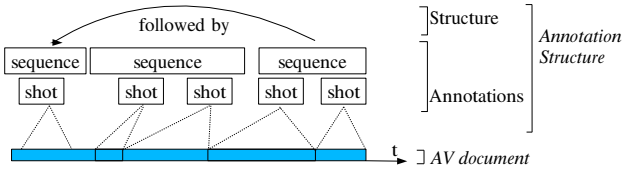

Fig. 2. An annotated audiovisual document

One of the design goals of Advene is to allow users to specify themselves how they want the annotations to be rendered. The Advene model defines the notion of *view*, that represents a way to display annotations. Moreover, visualising data also means selecting the data to be visualised: a *query* represents a way to select elements from the annotation structure.

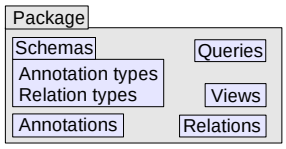

Fig. 3. Overview of the Advene model

Figure 3 gives an overview of the different elements of the Advene model. They are stored in documentary units called *packages*. A package is a document that holds all relevant information (schemas, annotations, queries and views) allowing to exchange, modify and visualise the metadata associated to an audiovisual document. Being separate from the audiovisual document, it can be modified and exchanged independently.

2.2 The Advene prototype

The Advene model is fairly generic. Some decisions regarding the implementation of query or view languages had to be made in the prototype.

Visualising annotations in the Advene prototype The Advene model defines a notion of view, without specifying what is in a view, which is a decision left to the implementation. The Advene prototype proposes three types of views: ad-hoc views (GUI views), static views (HTML templates) and dynamic views (set of rules allowing to dynamically modify the movie rendering).

Ad-hoc views are programmed views built in the GUI, that the user can configure. They feature standard views found in audiovisual software (time-line views, hierarchical data view, transcription view...).

Static views are XHTML templates that can be applied on the data. We are reusing the ZPT (Zope Page Templates) template system from the Zope platform [6]. This template system is oriented towards XML templates edition, using attributes in a dedicated namespace as processing instructions. Thanks to this attribute-based approach, both templates and result documents are valid XML documents, which allows us to process them with standard XML processing tools, like the *epoz* WYSISWYG browser-based editor [7] that has been integrated in the prototype.

Another component brought by the ZPT framework is the TALES syntax, that proposes a simple, path-like addressing scheme to address elements from a data model. This approach does not try to be a full query language, such as XPath wrt. XML, but instead to provide a simple, user-accessible way of addressing elements. For instance, the expression `/package/annotationTypes/sequence/annotations/first/content/data` addresses the content of the first annotation of type *diegetic sequence*.

Dynamic views are able to dynamically change the way the movie is played, based on the annotations' content. Using a rule-based model similar to the filtering capabilities of e-mail software (Event-Condition-Action [8]), dynamic views allow the user to specify various actions to be executed when some events occur. The actions range from simple VCR-like functionality (pause, go to a position, stop...) to more elaborate video control (display captions – text or graphic – on the video, get a snapshot...), and also provide user-interaction facilities (information popups, navigation popups offering to go to another position...). The events are triggered by the annotation structure (annotation begin, annotation end...) or by user actions (player pause, player start...).

With this simple rule-based specification, it is possible to enrich the movie with information issued from the annotation structure, or even change the way the movie is played. The rule *When the event **annotation begin** occurs, display the annotation content as a caption if the annotation type is **sequence*** displays the sequence title over the video. The rule *When the event **annotation end** occurs, go to the beginning of the related annotation if there exists a **followed by** relation* will make a dynamic montage of the movie, restoring the diegetic chronology of the sequences.

Queries offer a way to select elements from the annotation structure. A simple query implementation has been integrated in the prototype, using the same framework as the dynamic views: elements matching a given condition can be extracted from a given set of elements. This approach has proved flexible enough to accommodate various needs in our experimentations: selecting elements based on their contents, their temporal relationships (through Allen relations) or their relations.

Architecture of the Advene prototype The open-source Advene prototype reuses standard software components: it embeds the versatile, open-source and cross-platform VLC video player [9], uses the ZPT template model from Zope, and uses a standard web browser to visualise the rendition of the ZPT templates. Figure 4 provides an overview of the prototype architecture.

The Advene prototype has been written in python, which proved an excellent choice for rapid development and experimentation. It provides a testbed for the development

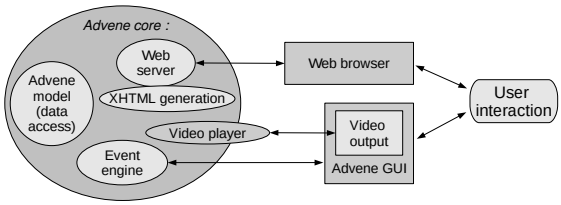

Fig. 4. The Advene prototype architecture

of new ideas in the field of multimedia annotation handling and visualisation. It is being used in ongoing collaborations with researchers in human interactions (who study video recordings) or movies study, as well as by individual researchers that use audiovisual material. The following section describes how we have integrated OWL in Advene.

3 Integrating OWL in Advene

In this section, we demonstrate how OWL descriptions and inferences integrate smoothly inside the Advene model presented before. This will be illustrated with the example Advene schemas from the previous section.

We propose here a two-steps integration of OWL in Advene: exposing Advene elements as an OWL description, then performing inference over it and getting the results back into the Advene model. The first step is achieved by using Advene *views*, while the second one is performed by dedicated Advene *queries*.

3.1 OWL views

Exposing Advene elements as OWL can be done in two non-exclusive ways: using views to "expose" Advene structures as OWL structures, or putting OWL statements inside annotations/relations (as their content).

Viewing Advene structures as OWL The first way is a direct application of the general notion of view in Advene, using OWL as a target format. It is straightforward in the current implementation since everything in an Advene package already has a URI, and since ZPT (the template language used to define static views) is able to produce any kind of XML document.

On the one hand, such views could do a straightforward "translation" of the Advene structure, according to an ontology of the Advene model (with classes such as Package, AnnotationType, Annotation, etc.). We actually designed such an ontology as a proof of concept, and a package containing the associated ZPT views[2]. The advantage of that package is that it can be imported in any other Advene package and enhance it with OWL export capabilities.

[2] `http://liris.cnrs.fr/advene/owl/1.0/advene-sw.xml`

On the other hand, schema authors may want to devise more specific OWL views in order to embed the underlying semantics of their schema, so that the produced OWL statements are more richly describing the annotation structure. For example, we can imagine that the designer of *diegetic chronology* would represent the binary relation *followed by* by an OWL property rather than by OWL instances, and impose that *diegetic sequence* annotations be followed by at most one other sequence.

We also envision that some Advene schemas could be designed on top of a pre-existing ontology in order to use that ontology in the context of video annotation. For example, one could want to describe the characters of a movie and the relationships between them by defining an annotation type *character* and a number of relation types corresponding to the properties in the *Relationship* ontology[3] (childOf, worksWith, enemyOf...), then provide a view converting annotations complying to this schema into an RDF description complying with that ontology. This scenario shows that, instead of considering the RDF description as a by-product of the Advene package, one can consider Advene as a front-end tool for annotating videos with RDF/OWL.

OWL statements in annotations/relations content Putting OWL statements inside annotations/relations is also a straightforward application of Advene principles, which does not impose any data type on their content. One could for example add such a content in annotations of type *shot* in order to formally describe the depicted scene (e.g. with the ontomedia ontology [10], intended to describe fictional films). One could then query each annotation individually to perform inference over its content. But inference would not here take advantage of the fact that annotations are attached to a fragment of the video stream.

Yet the anchoring of OWL statements in the stream can be taken into account by *grouping* several contents, according to various criteria which we call *reasoning contexts*, and which can in turn be materialized by other annotations. For example, one might want to reason on the content of all shots temporally contained in a given diegetic sequence. Or, assuming that a relation *appears in* exist between *character* and *shot* annotations, one might want to reason on the content of all shots where a given character appears (see figure 5).

Using annotations as reasoning contexts over multiple OWL-annotations can be achieved by defining specific views over the context annotations themselves, where an OWL ontology is generated, importing all the contents of the relevant annotations. TALES expressions and Advene queries can indeed be used in views to retrieve annotations based on temporal relationship or Advene relations. In the example of figure 5, applying the "temporal inclusion" view to $d1$ would generate an ontology importing the contents of $s1$ and $s2$, while applying it to $d2$ would import $s3$ and $s4$. On the other hand, a view using the *appears in* relation would generate an ontology importing $s1$ and $s3$ when applied to $c1$, and only $s4$ when applied to $c2$. We see that OWL statements can be used in different context, depending on the point of view used to group them.

[3] `http://vocab.org/relationship/`

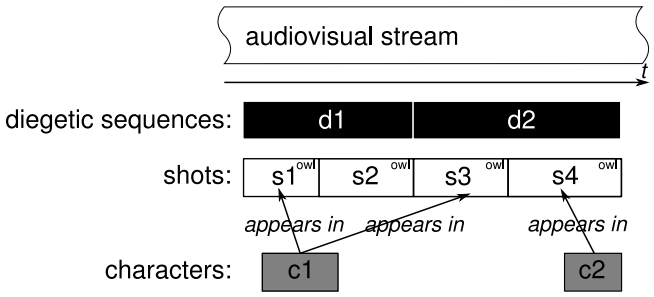

Fig. 5. Annotations of type *character* or *diegetic sequence* can be used as reasoning contexts for the OWL contained in *shot* annotations.

Mixing the two Of course, those two approaches can be mixed: specific OWL views can take advanrage of both the annotation structure and contents to provide rich descriptions of the annotated video, as described in [11].

3.2 OWL Queries

We just saw how Advene elements can be viewed through an OWL description. In principle, any DL inference service [12] can be used to query that OWL description. However, we mainly focused on the use of A-box querying (see section 4), for it integrates smoothly with the notion of query in Advene, as we will see.

For this purpose, we use the SPARQL language [4] to query the (deductive closure of) OWL views. More precisely, we restrict ourselves to SELECT queries[5]. The sample query in figure 6 illustrates the main features of SPARQL. The PREFIX clauses define namespace prefixes used in the other clauses. The FROM clause locates the source of information to be queried. The WHERE clause describes a subgraph to be searched, where some nodes (whose name starts with a question mark '?') are variables. Finally, the SELECT clause indicates which variables are to be returned. The result of such a query is a list of tuples (2-uples in our examples), each tuple being a binding of the selected variables, satisfying the query. We will now show how this is compatible with the notion of query in Advene.

Queries in Advene are used as *filters*: from a set of Advene elements (possibly the whole package), they select the subset of elements matching the query. SPARQL queries in Advene only requires that the initial set of items is described in RDF (which has been made possible by the OWL views described previously), and that the URI bound to the variables in the result are converted back to the Advene element they identify[6].

[4] `http://www.w3.org/TR/rdf-sparql-query/`

[5] SPARQL has other kinds of queries (CONSTRUCT, DESCRIBE) but they have different kinds of results, which do not fit in Advene queries

[6] Actually, there are two more slight differences: SPARQL queries return a set of *tuples* rather than a set of single elements, and those tuples may not only contain Advene element but

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rel:  <http://purl.org/vocab/relationship/>
SELECT ?x ?xn ?g
FROM   <http://movies.com/description>
WHERE  {
  ?x rdf:type    foaf:Person ;
     foaf:name   ?nx ;
     rel:enemyOf ?y .
  ?y foaf:name   "James Bond" .
  ?g rdf:type    foaf:Group ;
     foaf:member ?x .
}
```

Fig. 6. A SPARQL query, retrieving the URI and name of every enemy of James Bond belonging to a group, and the URI of their group.

Implementation We have implemented a basic SPARQL support in Advene using Pellet[7] as an external inference and query engine: Pellet is indeed able to perform A-box querying with SPARQL over OWL models. Pellet accesses the OWL views and the query through the HTTP server embedded in the Advene core (see figure 3).

4 Using OWL in Advene

In this section we explore the benefits, from the point of view of video annotation, of OWL-enhanced Advene. We do so by presenting a number of prospective scenarios made simple with OWL inference when they would have been complicated, when not unfeasible, with more "classical" queries and views. As illustrated in figure 7, some scenarios are focused on helping the annotator in her task, while others are focused on the end-user.

Checking consistency We already mentionned that OWL allows schema designers to express semantic constraints on the use of their schemas. The schema designer could easily state in OWL that, e.g., a diegetic sequence can not be directly followed by more than one sequence. OWL consistency checking can then be used by the annotator to ensure that her annotation structures complies with the underlying semantics of the schema. Some engines, including Pellet, even provide human-readable explanations of why a given OWL description is inconsistent. Indeed, inconcistency would probably lead to unsatisfactory results from the other views provided with the schema (assume a virtual montage restoring the diegetic order, which would not know how to choose between several following sequences).

also *strings* (literals and unresolved URIs). However, the current implementation of static and dynamic views has no problems dealing with tuples and strings.

[7] `http://www.mindswap.org/2003/pellet/`

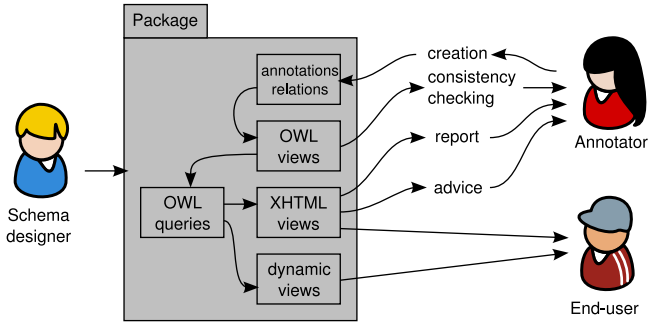

Fig. 7. The schema designer provides a schema (not represented) with OWL views, OWL queries, XHTML views and dynamic views adapted to that schema. The annotator creates annotations complying with the schema. Some of the view help her in her annotation task, while other views are aimed at the end-user.

Reporting Specific queries and static views can also be provided by the schema designer to the annotator, for her to check that everything complies with the intended semantics. A first advantage over plain consistency checking is that the "errors" can be more specificly explained in those views. Another advantage is that SPARQL queries can be more expressive than OWL classes. Let us consider our example using diegetic sequences and shots, where the OWL content of shots uses an ontology describing characters and events involving them. A SPARQL query could be used for finding all sequences temporally containing a shot describing the death of a character, and diegetically followed (directly or indirectly) by a sequence containing a shot involving that character[8] (note that such a query requires at the same time OWL representing Advene structures and OWL in annotations content, see section 3.1). A dedicated static view can then report all *post-mortem* occurrences of a characters to the annotator. A third advantage, and an important one, is that such reporting views provide a *finer grain* than boolean consistency: it can be only *suggested* that characters should not resurrect, but some movies might not comply with that constraint.

Advising Moreover, some static views could even *advise* completions or modifications of the structure or content of Advene elements, based on *annotation patterns* expected by the schema designer. For example, a static view could generate an XHTML form proposing to add *followed by* relations in an incomplete diegetic chronology, possibly excluding orderings where characters would reappear after their death[9].

Generating more hypermedia We just saw how static views can be generated thanks to OWL inference for helping annotators in their task. The same mechanism can of

[8] This can not be expressed as a class because it implies a cycle in the graph. However, the SPARQL uses inference to take into account transitivity of the *followed by* relation.

[9] The HTTP server of the Advene core can indeed modify the data model in response to dedicated GET or POST queries.

course be used to generate hypermedia aimed at end users. Non-trivial queries can be used to extract a set of elements (e.g. “all shots representing characters who are resurrecting at some point of the movie”), then use them to generate an appropriate static view (list of the shots, snapshots of such characters appearing) or dynamic view (virtual montage with only those shots, in the original or diegetic order).

4.1 Discussion and related work

Despite the long acknowledged need for semantically annotating multimedia documents, the unification of multimedia annotation standards with Semantic Web technologies is still a work in progress. An alleged difficulty for this unification is the lack of interoperability between standards [5, 13], especially between XML-based MPEG-7 and RDF-based OWL. Interoperability has however to be achieved since Semantic Web technologies “as is” are not quite adapted to multimedia annotation —though some approaches attempt to fill that gap [14]. Various approaches have hence been proposed, either to convert MPEG-7 structures into RDF based language in order to be able to reason about them [15], or to embed OWL ontologies into MPEG-7 structures in order to take advantage of standard-compliant tools, while retaining the semantics of the description [16]. While the former may be compared with the first approach presented in section 3.1, the latter can be compared to our proposition (in the same section) of designing Advene schemas according to existing ontologies.

Another hindrance to the large adoption of multimedia annotation in general is the complexity of the dominant standard MPEG-7 [5]. It is interesting to note that Semantic Web annotation is often the target of the same criticism, as working with (sometimes big) formal ontology requires some training for unskilled users. Advene eschews both by relying on a simple and extensible model for video annotation, and by not relying on formal ontologies from the bottom; we rather propose to use third-party or ad-hoc OWL ontologies on an opportunistic basis, i.e. when (and if) they can prove useful in a given context. By doing so, we argue that Advene meets the requirements for practical multimedia annotation expressed by [17].

Finally, the Advene architecture can provide the functionalities targeted by other approaches: controlling and checking the structure of annotations [16] as seen in section 4, semantic information retrieval [14] thanks to OWL queries, virtual montage [18]. But advantage can also be taken from semantic annotation by other uses, such as enriched video viewing, which are not, to our knowledge, addressed by this community.

5 Conclusion

In this article we have presented some ideas for integrating semantic web technologies in an annotation-based hypervideo system. The very simplicity of the advene model, and the versatility of the advene prototype made it easy to propose numerous ideas, backed by a preliminary implementation. Current work on the “semantic web side” of the advene project entails smoother integration of OWL-queries in the prototype and graphical editing of such queries, design of OWL-views for consistency checking and reporting, design of reasoning-enriched dynamic views, and theoretical study of

the notion of “reasoning context”. The advene prototype is freely downloadable and extensible, and we encourage anybody to use it for testing new ideas on multimedia and semantic web.

References

1. Aubert, O., Prié, Y.: From video information retrieval to hypervideo management. In: Corimedia, the international workshop on multidisciplinary image, video, and audio retrieval and mining, Sherbrooke, Canada (2004) 10 pp.
2. Aubert, O., Prié, Y.: Advene: active reading through hypervideo. In: ACM Hypertext'05. (2005)
3. Sanchez, J.M.M., Koenen, R., Pereira, F.: MPEG-7: The Generic Multimedia Content Description Standard, Part 1. IEEE Multimedia Journal **9**(2) (2002) 78–87
4. Pfeiffer, S., Parker, C., Schremmer, C.: Annodex: a simple architecture to enable hyperlinking, search and retrieval of time-continuous data on the web. In: 5th ACM SIGMM International workshop on Multimedia information retrieval. (2003) 87–93
5. van Ossenbruggen, J., Nack, F., Hardman, L.: That obscure object of desire: Multimedia metadata on the web, part 1. IEEE MultiMedia **11**(4) (2004) 38–48
6. Zope Corporation: Zope Page Templates reference. (2004) `http://www.zope.org/Documentation/Books/ZopeBook/2_6Edition/AppendixC.stx`.
7. Jablonski, M.: Epoz, a cross-browser WYSIWYG editor for Zope. (2003) `http://epoz.sourceforge.org/`.
8. Paton, N.W., ed.: Active Rules in Database Systems. Springer Verlag, New York (1999)
9. Fallon, H., de Lattre, A., Bilien, J., Daoud, A., Gautier, M., Stenac, C.: VLC User Guide. VideoLAN Project. (2003)
10. Lawrence, F., Tuffield, M.M., Jewell, M.O., Prügel-Bennett, A., Millard, D.E., Nixon, M.S., Schraefel, M., Shadbolt, N.R.: OntoMedia - Creating an Ontology for Marking Up the Contents of Heterogeneous Media. In: Proceedings of Ontology Patterns for the Semantic Web ISWC-05 Workshop, Galway, Ireland (2005)
11. Troncy, R.: Integrating Structure and Semantics into Audio-visual Documents. In: Second International Semantic Web Conference (ISWC2003), Sanibel Island, Florida, USA, Springer (2003) 566–581
12. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
13. van Ossenbruggen, J., Stamou, G., Pan, J.Z.: Multimedia Annotations and the Semantic Web. In: Proc. of the International Workshop on Semantic Web Case Studies and Best Practices for eBusiness (SWCASE). (2005)
14. Isaac, A., Troncy, R.: Using several ontologies for describing AV documents : a case study in the medical domai. In: 2nd European Semantic Web Conference, Workshop on Multimedia and the Semantic Web, Heraklion, Crete (2005)
15. Hunter, J.: Adding Multimedia to the Semantic Web - Building an MPEG-7 Ontology. In: International Semantic Web Working Symposium (SWWS), Stanford (2001)
16. Troncy, R., Carrive, J.: A reduced yet extensible audio-visual description language. In: Proceedings of ACM Document Engineering. (2004) 87–89
17. Geurts, J., van Ossenbruggen, J., Hardman, L.: Requirements for practical multimedia annotation. In: Workshop on Multimedia and the Semantic Web, Heraklion, Crete (2005) 4–11 part of 2nd European Semantic Web Conference.
18. Bocconi, S., Nack, F., Hardman, L.: Supporting the generation of argument structure within video sequences. In: ACM Hypertext'05. (2005) 75–84