The Advene Model for Hypervideo Document Engineering

Olivier AUBERT, Pierre-Antoine CHAMPIN, Yannick PRIÉ
LIRIS FRE 2672 CNRS - Lyon 1 University
69622 Villeurbanne Cedex

2004-05

Abstract

The Advene (Annotate DVd, Exchange on the NEt) project[1] is aimed towards communities exchanging discourses (analysis, studies) about audiovisual documents (e.g. movies) in DVD format. This requires that audiovisual content and hypertext facilities be integrated, thanks to annotations providing explicit structures on audiovisual streams, upon which hypervideo documents can be engineered.

The Advene framework provides models and tools allowing to design and reuse annotations schemas; annotate video streams according to these schemas; generate and create Stream-Time Based (mainly video-centred) or User-Time Based (mainly text-centred) visualisations of the annotations. Schemas (annotation- and relation-types), annotations and relations, queries and views can be clustered and shared in units called packages. Hypervideo documents are generated when needed, both from packages (for annotation and view description) and DVDs (audiovisual streams).

This article is mainly dedicated to introducing the various elements of the Advene model, illustrated with a use scenario, and the Advene prototype, composed of an augmented video player with DVD capabilities for Stream-Time Based Visualisation, while user time based visualisation takes place in a standard web browser, thanks to the XML based template language TAL.

Keywords: Annotation, Document template, DVD, Sharing, Video, Visualisation modes, Advene

1 Introduction

Digital audiovisual (AV) documents are getting more and more common on our computers. They are being used in various domains, such as news, entertainment, videoconferencing, surveillance, teaching, etc. Moving images are being streamed across the Internet, digital TV is getting interactive, and fictions, documentaries, or many kinds of live performances now benefit from a digital diffusion on DVD medium.

[1]A work done at LIRIS: Lyon Research Center for Images and Intelligent Information Systems, FRE 2672 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/Ecole Centrale de Lyon

In the same time, digital video data is more and more used in the humanities [AP99] and video analysis systems (e.g. [Kip01]) are being designed for humanities corpus study (e.g. sociology, linguistics, speech acts, ...) or sport analysis.

The context of our work is related to working with movies on DVD medium and sharing this work inside communities: filmic analysis for film enthusiasts or researchers, use in classroom for teachers and pupils (e.g. in movie, foreign language or literature domains). We want to consider uses of video documents that are different from simple visualisation and consumption. This means that these uses rely on interfaces for using extended AV documents (hypervideos[2]), whose characteristics reside in the possibility of interacting with the stream in a non trivial way. This also means that users should be able to annotate the streams and build their own hypervideos.

It then becomes necessary to have structured descriptions of audiovisual material — that should be set up during an annotation process —, and tools enabling their edition, as well as their reading/visualisation. Document models are needed, that permit first to localise fragments that are relevant for a certain analysis (hence for a certain use based on this analysis); second to qualify them (hence to attribute semantics to them, and a place in the structure of a document model); finally to use them in graphical interfaces using the document structures related to the video stream.

Although in past years numerous systems, offering audiovisual description models and their associated prototypes, have been designed, they are generally limited to specific domains and precise uses (e.g. gestural analysis, transcription...). Innovative uses of hypervideos are currently emerging, but no widely used killer application for digital (hyper)videos seems to exist, nor do attested widespread usages (if we except subtitling), and almost everything remains to be invented.

We think that the development of new usages (hence of new document models) should be based upon user communities. The Advene project (Annotate DVds, Exchange on the NEt) relies on this assumption, and our goal is therefore to provide tools for *active reading*[AP99] and work on DVD-based audiovisual documents. These tools should, among other things: be adapted to communities having interest in exchanging documentary descriptions; permit the production of new documents from others; allow the assisted analysis of AV documents.

Furthermore, audiovisual right is very restrictive. Hence, we consider it vital to separate the description of AV documents from the actual streams so that it is possible to work on the description without the streams (e.g. for information retrieval), and to let dissemination of streams be independent of their descriptions. The medium we have chosen for our project (DVD) allows us to manipulate high quality video documents (fiction, documentaries,...), around which communities can structure themselves, while reading rights are acquired by the actual readers of the streams.

Involving user communities and taking into account audiovisual rights are two primary concerns of the Advene project. We have identified several functionalities that our system should provide (and that our model for describing AV documents and designing hypervideos documents should support):

[2]The term *hypervideo* comes from the Hypercafe experimentation [SBS96], but we will use it in a extended way: it will denotate both videos with hypertext capabilities, and hypertext documents with video controls, into which video plays a non anecdotic role.

- editing annotations of various types, that are related to the AV stream, and linked by relations, forming a structure that can be used for hypervideo rendering;

- editing *annotation schemas*, ie. modelling the structure imposed on the annotation, hence the hypervideo document structural part (the other part being the modelling of hypervideo visualisation with views);

- visualising hypervideo documents built from the AV stream and the annotations (through views), constructed from templates that can be reused or specifically designed. Indeed, there are two main ways of considering the visualisation of an audiovisual document extended with an annotation structure (or of an annotation structure together with the AV stream it is related with): yielding to the temporality of the stream, or spatialising (delinearising) it. In the first case, the visualisation will concentrate on a video player with hypervideo annotation-based capabilities (*Stream-Time Based Visualisation*, noted STBV). In the second case, it will mainly rely on standard hypertext representation of the annotations together with standard hypertext content, with limited video capabilities (*User-Time Based Visualisation*, noted UTBV).[3] As a first approximation, we can consider STBV as annotation-enhanced video and UTBV as annotation-based generation of HTML documents.

- tools and models for querying the database composed of the annotation structure, but also of the annotation schemas, the visualisations, etc.

- tools and models for sharing the annotations and the hypervideos document they allow to visualise.

Figure 1 presents an overview of the annotation process: using a given source (DVD) and the appropriate (i.e. dedicated to the task at hand) schema, the user creates supplementary information in the form of annotations and relations. As a second step, hypervideos are generated by views that use the video source and the edited annotations as data.

This article is dedicated to presenting the Advene model and architecture as a powerful way of creating, engineering and sharing hypervideo documents. The following section (second) deals with the architecture and the design needs of the Advene framework. The third section present the Advene model. The working prototype that we developped is described in section four, with screenshots and some details about the ongoing experiments. In the fifth section, related works are compared to our approach along several aspects.

2 Scenario and needs

We present an example use scenario of our framework, based upon real experiments, upon which we will explain various points. Miss X, a foreign language

[3]This distinction can be related to the one between "video centred pages" and "text centred pages" in [CG02]. In STBV, the temporal reference of the resulting hypervideo is mainly given by the (original) video stream time. In UTBV, the temporal reference of the resulting hypervideo is more dependant of the user actions (mainly hyperlink activation), hence of his time.

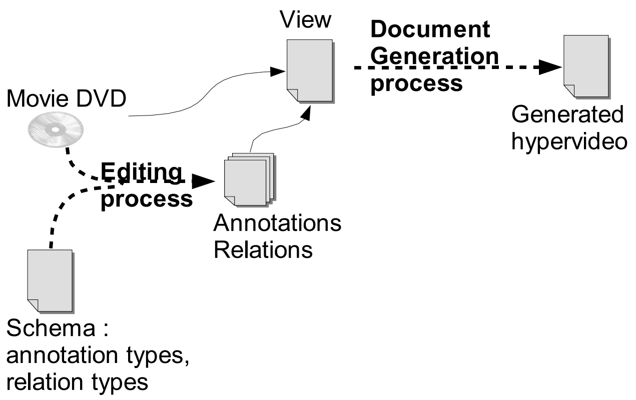


Figure 1: Process overview

teacher wants to use a movie as a pedagogical exercise. Her class consists in two steps:

- Have the pupils watch the movie by sequences. While the movie plays, watching tips are displayed on the screen (important vocabulary, grammar points, ...). At the end of each sequence, the video player automatically stops.
- Give an exam about the movie. The questions are illustrated by some screen captures of the movie, and the important vocabulary is given again on the exam sheet.

To prepare this class, Miss X uses the Advene prototype. She has to define a structure of annotations relevant for her pedagogical activity, annotate the movie according to this structure and eventually use the result with her pupils.

An educational institution provides on its web server *annotation schemas* designed for this kind of activity. Each schema defines various *annotation types* and *relation types*, related to the activity (denoting a specific dimension of analysis). For example, the *watching-tips* schema defines the following annotation types: *character* (annotating a meaningful occurrence of a character), *vocabulary* (annotating the occurrence of a difficult word, with its definition), *caption* (holding subtitles for the studied movie).

Generic *views* are also bundled with the schemas, displaying some of the annotations in a meaningful way. For instance, a User-Time Based View *vocabulary-list* produces a hypertext document listing all the difficult words of the movie, and allows to visualise the moment of occurrence of each word. A Stream-Time Based View *with-watching-tips* produces an augmented playback of the stream with subtitles and visual cues added (from the *watching-tips* schema) to the original movie.

Mr Y, one of Miss X's colleagues, already studied the movie in his class and thus created vocabulary annotations and defined video sequences, using the same schemas. He gives her the XML file holding his annotations, for her to reuse. They do not have to exchange DVDs, both of them having their own copy.

Using the same schemas as Mr Y, Miss X can reuse, by copying or referencing, some of his annotations. She can add her own annotations (for instance, vocabulary more relevant to the level of her class). She can even design and use her own schema, corresponding to a kind of exercise not covered by the institution's schemas. Finally, she writes another document as a UTBV, illustrated with movies screenshots and annotation contents, that she will use as an exercise sheet for her pupils.

Figure 2 represents the different steps of the preparation process described above.

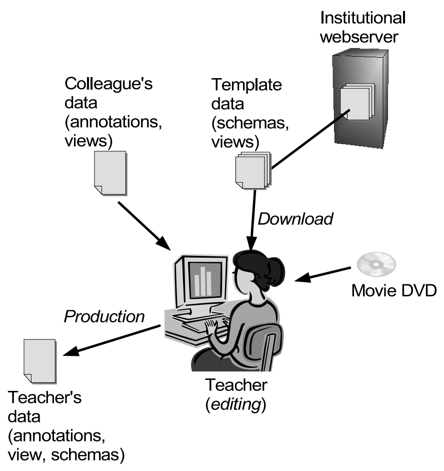

Institutional webserver
Colleague's data (annotations, views)
Template data (schemas, views)
Download
Movie DVD
Production
Teacher's data (annotations, view, schemas)
Teacher (*editing*)

Figure 2: Editing scenario

At the beginning of the class, the teacher distributes a sheet with the difficult vocabulary list, generated by the generic UTBV *vocabulary-list* from the institution's schemas, so that the pupils can have a written support.

Then the pupils watch the STBV *with-watching-tips* which displays the movie enriched with the *watching-tips* annotations and stops at the end of each sequence. Finally, the teacher distributes the exercise sheet that she wrote, enriched with annotation data and movie screenshots.

Figure 3 represents the different steps of the use scenario described above.

From this simple scenario, we can identify some of the issues that our framework has to deal with.

First, the user must have the ability to reuse existing annotation schemas as well as define her own. The reuse of existing schemas enables the user to share her annotations with other users. The possibility to define additional schemas does not confine the user to a closed set of schemas (i.e. of analysis dimensions), but allows her to express the most relevant information in her analysis.

Second, beyond schema reusing, data sharing is a crucial issue, as it provides the opportunity to build upon existing work. Data sharing may involve any kind of element (schemas, annotations, views...). It can be done by copying data, or by referencing it.

Third, we are aiming at providing tools to a large community. The wide use of the software depends on its easy installation and use. It especially should

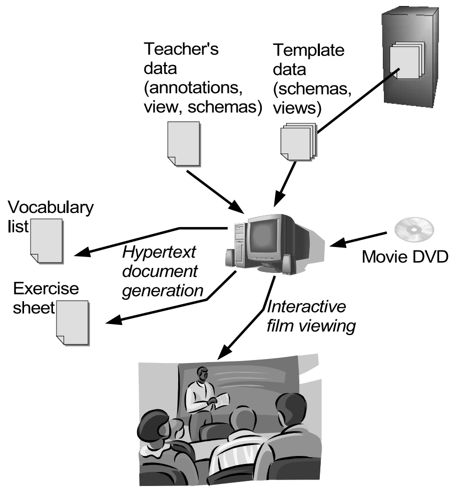

Teacher's data (annotations, view, schemas)
Template data (schemas, views)
Vocabulary list
Hypertext document generation
Exercise sheet
Movie DVD
Interactive film viewing

Figure 3: Use scenario

run on personal computers with common software and hardware.

Finally, as mentioned in our introduction, such a scenario shows the necessity of two modes of visualisation. The first one consists in viewing the movie with accompanying annotations (STBV), while the other one, as the exercise sheet, relies on the annotation data as primary base (UTBV).

In the next section, we will see how the proposed model addresses these issues. The following section will present the status of the prototype that already implements a great number of the functionalities discussed in the scenario.

3 The Advene model

Advene aims at providing a simple yet extensible framework. The focus has been put on the *external* structure of annotations, *i.e.* how they are constrained, related with each other, and rendered as hypervideos. Indeed, annotations are not considered as undistinguishable pieces of data randomly linked to audiovisual documents. They are structured, according to *annotation schemas*, which are specific to a particular annotation task. Groups of related annotations are therefore bundled with their schemas and views for rendering them, the whole being called a *package*.

On the other hand, as few assumptions as possible have been made on the *internal* structure of annotations, *i.e.* what kind of data is used to annotate audiovisual documents (plain or formatted text, XML, audio...). Advene relies on externally defined content types and languages, as well as a plugin mechanism in order to edit and render such data.

3.1 Overview

The central notion of the Advene model is the notion of *package*. A package is a relevant (from its author's point of view) set of Advene elements grouped together, in order to be stored and shared as a whole. Figure 4 shows a UML

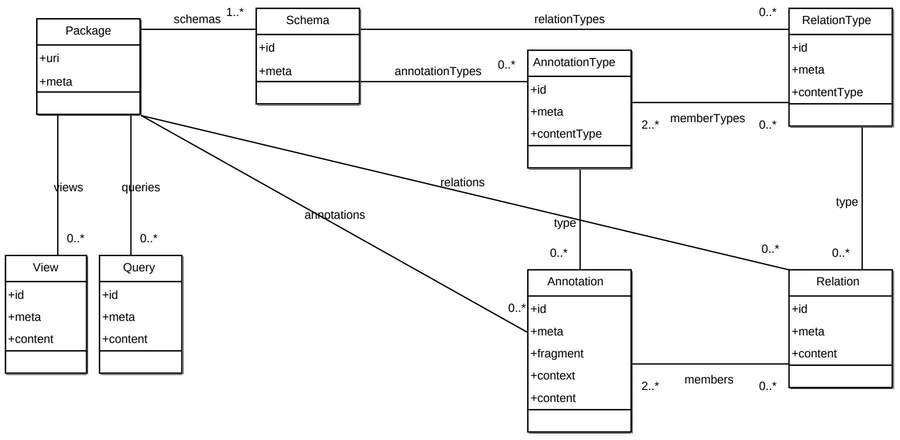


Figure 4: UML diagram of the Advene model

diagram of the different classes of elements one can find in an Advene Package, and the associations between them. Annotations and Relations between them hold the data annotating audiovisual documents. All annotations are not equivalent (nor are the relations): they all belong to a particular *type*, AnnotationType or RelationType, belonging in turn to a particular Schema from the same package. To manipulate and present the data, packages also define queries and views. Queries are used to select a subset of the package's elements. Views produce new multimedia documents by combining parts of the annotated audiovisual medium and components of the package.

Defined and imported elements It is worth noting that this diagram is actually simplified. Each component of a package can actually be either defined inside the package itself (*i.e.* explicitely described), or imported from another package (*i.e.* described only by referencing its real description in that other package). Similarly, a schema can either define or import AnnotationTypes and RelationTypes. By providing a unified access to defined and imported elements, packages ensure an easy exchange and sharing of all the necessary information[4] to generate hypervideos from annotations.

3.2 Model elements

3.2.1 Common features

All the components of packages are identified by a name, denoted *id*. This identifier is bound to be unique in the package *defining* the component (in opposition to packages importing the component, *i.e.* making a reference).

All the components of the model (annotations, relations, schemas, views, ...), as well as the package itself, have additional attributes described by a set of metadata, named *meta*. Any number of metadata schemes can be used in

[4] The AV stream itself is *referenced* by each annotation, although not contained in the package.

this set. The Advene framework strongly encourages the (non exclusive) use of the Dublin Core metadata element set[5]. For each component described below, a number of recommended Dublin Core fields is specified.

3.2.2 Packages

A package is the grouping in a single document of all the Advene elements (schemas, views, annotations, etc.) produced or used during the active reading of one (or potentially several) DVD, in order to ease the use and sharing of these elements. It is identified by a URI. The author and creation date of the package should be represented as metadata using the Dublin Core vocabulary.

A package also holds a list of all packages from which it imports components (which is not represented in the diagram). Among other good properties, this enables a clear statement of dependancies between packages. Another important role of this list is to assign a *namespace identifier* to each imported package; this enables to prevents name clashes components from different packages, by prefixing imported component identifiers with the namespace name, similarly to what is done with XML namespaces [BHL99].

3.2.3 Annotations, Relations

Annotations are pieces of data (the *content* of the annotation) linked to a *fragment* of the audiovisual document. It is worth noting that identifying a fragment in an audiovisual medium depends on the specific structure of that medium. In standard digital movies, only timepoints or bytecounts can be addressed. DVDs, on the other hand, are structured in titles and chapters. The MPEG-4 standard even enables the addressing of spatio-temporal fragments. The Advene prototype currently defines a fragment as the interval between two timepoints, but is extensible to other fragment types.

The DVD medium offers a number of visualisation options: the same movie can be watched in different languages, with or without subtitles, even sometimes with different camera angles. Those parameters define a visualisation *context*. An annotation defined in a particular context will not always be relevant in other ones: for instance, an annotation about the quality of the french dubbing is no more relevant when the movie is watched with the english soundtrack. In the Advene prototype, the context of an annotation may optionally specify the soundtrack, subtitles and angle for which the annotation is relevant. However, as for fragments, the Advene model is extensible with other context types.

By definition, annotations are in relation with the audiovisual medium, by means of their `fragment`. It can also be useful to express the fact that annotations are in relation with one another. For that purpose, Advene has a Relation class, which is not directly linked to the video, but is linked to several annotations (the *members* of the relation). Relations may also have a content, in order to provide additional information.

A package can define its own annotations and relations, and import some from other packages. However, each annotation and relation must match one of the package's schema (see below).

[5]`http://dublincore.org/`, Dublin Core Metadata Initiative

3.2.4 Types

All annotations or relations are not equivalent. AnnotationTypes and RelationTypes identify and constrain different kinds of annotations or relations.

Annotation types have a *contentType* defining the kind of content their annotations may have. This is achieved in the Advene prototype by means of a MIME type [FB96] (`text/xml`, `audio/*`, etc.). If the type is `text/xml`, it can be more precisely constrained by, *e.g.*, a DTD, XML Schema or Relax NG schema[6]. For basic needs, a simple, non-constrained text content is used. For more elaborate needs, the content can be a document conforming to a specific XML schema, such as the definition of a thesaurus in MPEG-7.

Relation types define the number of members, *i.e.* the number of participating annotations, of the corresponding relations, as well as the annotation types to which the members must belong (via the `memberTypes` association). Since relations may have a content, relation types also have an optional `contentType`, similar to the one in annotation types.

3.2.5 Schemas

Annotation *Schemas* constrain the kind of annotations and relations that a package can contain. More precisely, it defines a collection of annotation types and relation types. Those types are grouped on the basis of the task they help to perform. For example, annotation types useful for editing a movie will be grouped in a given schema (*e.g.* annotation types Shot and Sequence, relation type Transition); another schema will contain types useful for analyzing filming techniques (*e.g.* Camera Move, Lights...); a third one will contain useful types for showing the movie in a language class; etc.

Note that this grouping is not exclusive, though: a schema can import types from other schemas. For example, the annotation type Shot, defined in the Edition schema, will also be used in the Filming Techniques schema, in order to point shots where some camera moves occur.

Note also that schemas do not have to be highly structured. The Advene prototype provides a very simple schema named SimpleText, providing a single annotation type, with plain text content. We envision that many annotation tasks would start with this "note taking" schema, should the annotations be further converted into more structured schemas with semantically richer types.

3.2.6 Queries

A Query is used to get a list of Advene elements (of either type described here). It can be given a number of parameters. The *content* of the Query object describes the query in a query language identified by its MIME type. The Advene prototype only implements Python as a query language for the moment (a straightforward implementation since the prototype is written in Python). More user-friendly query languages will have to be implemented in the future.

Queries can be defined by an end user as a convenient way of accessing a relevant collection of elements from the package. However, their use becomes more valuable combined with the import mechanism of packages: a package

[6] Although XML is the most popular data format providing additional mechanisms for constraining its instances, the same could be applied to any other data type providing a similar mechanism.

defining a number of reusable schemas (like the one on the institutional Web server in the previous scenario) will usually provide, in the same package as the schemas, a number of queries which are relevant to annotations conforming to those schemas. When importing the schemas, users will import the queries as well and get a full schema-specialized "toolbox" enabling them to handle and view their annotations in useful ways.

3.2.7 Views

Views are the final components in the chain enabling to produce various hyperdocuments from Advene packages. As described above, queries are used to filter and select packages elements. Views are then used to render those elements under various forms, either Stream-Time Based or User-Time Based, as discussed earlier in the paper. Conforming to the genericity goal of Advene, views can be described in any formatting language identified by its MIME type. The Advene prototype currently implements two view languages, which will be described and discussed later in this paper (section 4.2 and 4.3).

As with queries, reusable schemas will probably be provided together with dedicated views for rendering, e.g., annotations of the types they define. Specific views can also be defined by end users in their packages.

3.3 Packages and documents

We conclude here this presentation of the Advene model by demonstrating how it can be considered as a *document model*.

The Advene package is in itself a document: it is a consistent set of information about one or several AV streams, written by an author with a given purpose. It can be refered to (e.g., by another package), and exchanged.

The Advene package with the DVD is also a multimedia hyperdocument, resulting from the rendering of all its views. This document is multimedia since it can be composed of text, images or AV extracts, and can be either User-Time Based or Stream-Time Based. It is also a hyperdocument since it can contain links from one part to another, or links to external resources. It is worth noting that some views may not require the DVD at all to be rendered.

Finally, the rendition of a particular view can also be seen as a standalone document (e.g., the exercise sheet in the scenario from section 2). Views are not limited, in their goal, to what one usually expects from a style sheet. For example, a virtual video document [LV98] can be generated on the fly by imposing the structure of annotations to the original stream (e.g. following the narrative chronology). A structured discourse about an annotated movie, using annotation contents or snapshots, is also considered as a view of the corresponding package. An important part of the content of such a discourse may not come from the annotations but from the view itself. Indeed, such views should rather be thought of as *document templates* than styles sheets. This demonstrates the complementarity of views and annotations in the production of documents from an Advene package.

4 The Advene prototype

In this section, we present a global overview of the Advene prototype, before presenting in more details the two implemented view mechanisms, based on the TAL description language and on the ECA paradigm. Eventually, we detail the status of the prototype and its current and future uses.

4.1 Global software architecture

The global software architecture of the Advene prototype is shown in figure 5.

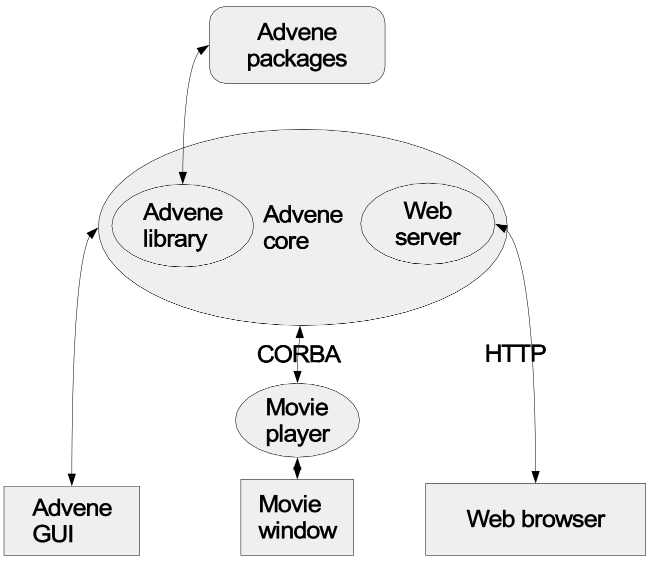


Figure 5: Overview of the prototype architecture

A library has been designed to ease the access to the data representing Advene packages and stored in XML files. It provides high-level programming elements such as Packages, Schemas, Annotations, etc. The applications (the Advene interface and the integrated webserver) access the package's data through this library.

A software DVD player provides the movie playback functionality. The Advene framework can remotely control the player[7]. Among the control possibilities are the display of textual data on the video display, and the ability to capture low-resolution screenshots of the movie.

Annotations are created and edited through the Advene GUI during the movie playback. The current version only allows textual annotation editing, but the underlying architecture does not restrict the annotation types in any way, and we plan to provide a plugin architecture to deal with other types of data (for instance, an audio recorder to record audio comments).

[7] This has been achieved by extending an existing DVD player with a CORBA control plugin. Using CORBA brought remote control, language independance as well as the definition of a precise interface to interact with the player. Supporting another DVD player only necessits to make it provide the same interface.

Editing assistants can also help in the process of annotation editing. For instance, an assistant can generate a set of annotations based on the chapter/title structure of the DVD, in order to give a working base. Building on previous work in the field of image processing, a more sophisticated assistant could autonomously detect scene changes in the movie, and generate annotations based on this information, that the user could then edit to suit her needs.

Once the annotations are created, the user can access them through the interface, for instance by playing the movie and having some annotations displayed as captions on the movie display (Stream-Time Based View).

The Advene framework also integrates an embedded webserver, which provides access to the package's data through a standard web browser, dynamically generating hypertext document from the views defined in the package (or imported packages).

These views (User-Time Based Views) are generated by the TAL engine of the Advene software, which integrates Advene elements into view templates. Image inlining is achieved by generating a specific URL which is queried from the embedded Advene webserver. Upon receiving a query for such an URL, the Advene software dynamically makes a screenshot for a given frame of the movie. For performance reasons, an in-memory cache has been implemented in order to alleviate the access times.

As the webserver is embedded in the Advene core, it also allows the user to control the movie player through the web browser.

Figure 6 shows a screenshot of the application, demonstrating different views, and involving the main Advene prototype components: the enhanced video player and the embedded web server. The displayed example is taken from the use scenario presented in section 2, a pedagogical use by a language teacher.

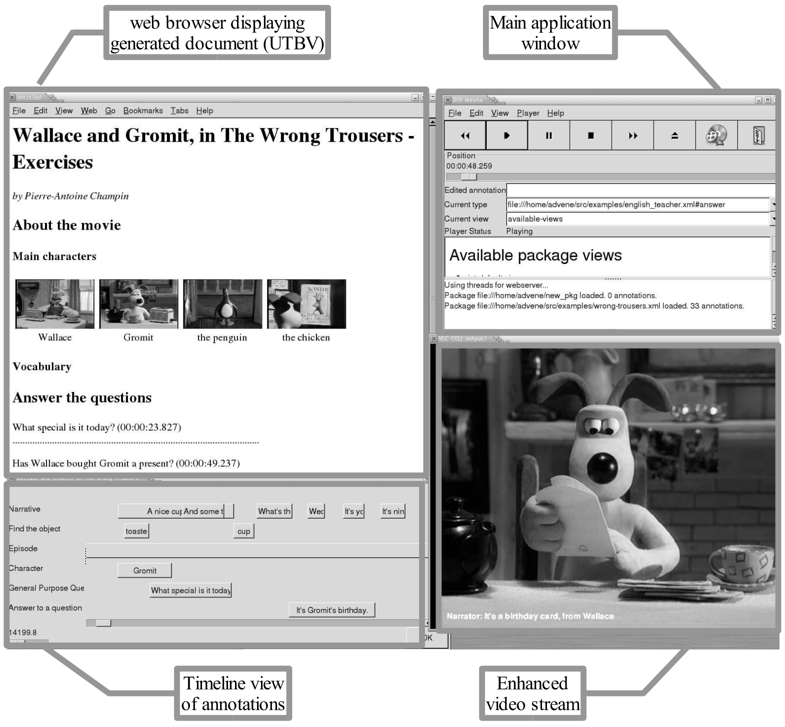
4.2 Hypertext view based on TAL

As said before, any formatting language, such as XSLT, can be used in Advene. However, for our prototype, we have chosen to implement another formatting language, named TAL (*Template Attribute Language*). It has originally been designed for the Zope platform[8] as a language for producing dynamic HTML documents, but is not limited to this purpose since it can be used to produce any kind of XML document. We will first present how it works, then explain why we have chosen it.

Principle The principle of TAL is to define a *template* document, i.e. a document very similar to the final one (the HTML or XML document to be produced). The main difference in both documents is the occurrence, in the template, of special XML attributes from the `tal:` namespace. These attributes are used as processing instructions (replacement, iterative, conditional, etc.) by a TAL engine, which transforms the template into the final document. The use of a dedicated namespace implies that the template documents are still *valid* XML documents, with only extra attributes added. Thus we can produce and manipulate them with standard XML tools (GUI or scripting).

The values of the `tal:` attributes are the parameters to the TAL instructions. They are expressed using an extensible syntax named TALES (*TAL Ex-*

[8] http://www.zope.org/Wikis/DevSite/Projects/ZPT/TAL



Wallace and Gromit, in The Wrong Trousers - Exercises

by Pierre-Antoine Champin

About the movie

Main characters

Vocabulary

Answer the questions

What special is it today? (00:00:23.827)

Has Wallace bought Gromit a present? (00:00:49.237)

Position 00:00:48.259

Edited annotation
Current type file:///home/advene/src/examples/english_teacher.xml#answer
Current view available-views
Player Status Playing

Available package views

```
Using threads for webserver...
Package file:///home/advene/new_pkg loaded. 0 annotations.
Package file:///home/advene/src/examples/wrong-trousers.xml loaded. 33 annotations.
```

What special is it today
It's Gromit's birthday.

Narrator: It's a birthday card, from Wallace

Figure 6: Advene screenshot

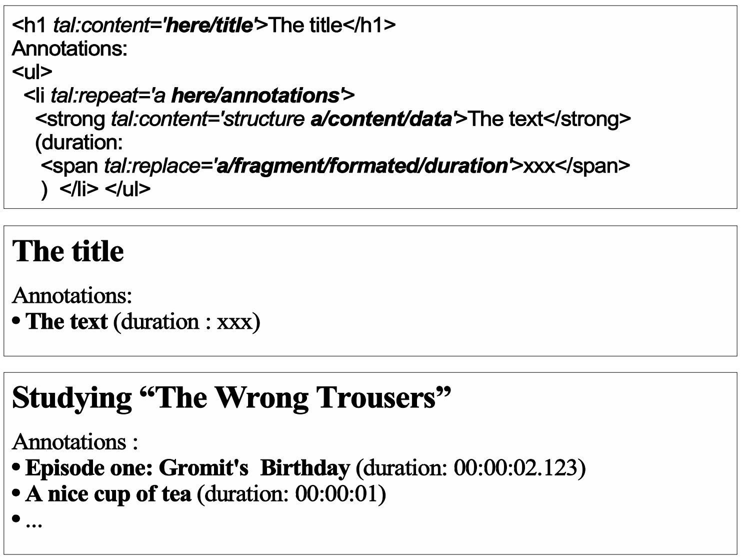
pression Syntax). TALES expressions look like pathnames inside a hierarchical view of the data. For instance, the duration of the fragment linked to the first annotation in the package is expressed as `/package/annotations/first/fragment/duration`.

Figure 7 shows an example in HTML with its source, how the template would be rendered if passed directly to a browser, and how the final document is rendered.

TAL is mostly used to produce HTML documents, i.e. User-Time Based Views in the Advene terminology. However, TAL has the ability to produce any kind of XML document. It is therefore possible to generate SVG, SMIL or MPEG-7 documents, to interoperate with other applications or to use as another output format.

Rationale We have chosen to implement TAL/TALES instead of other formatting languages (the first of them being XSLT combined with XPath) for a number of reasons.

First, we want to explore the capabilities of TAL and analyse the reasons of its success in the context of the Zope platform. It does not claim to be as comprehensive and powerful as XSLT, but tries to make it simple to achieve simple tasks. The targeted audience of Advene being non-expert end-users, simplicity is very important. The template-based nature of TAL is also an advantage in this area: it was designed to allow designers to compose page templates with their favourite WYSIWYG editor, giving them a good overview of the rendered result. The fact that TAL documents are *valid* XML documents



```
<h1 tal:content='here/title'>The title</h1>
Annotations:
<ul>
  <li tal:repeat='a here/annotations'>
    <strong tal:content='structure a/content/data'>The text</strong>
    (duration:
     <span tal:replace='a/fragment/formated/duration'>xxx</span>
     )  </li> </ul>
```

The title

Annotations:
- **The text** (duration : xxx)

Studying "The Wrong Trousers"

Annotations :
- **Episode one: Gromit's Birthday** (duration: 00:00:02.123)
- **A nice cup of tea** (duration: 00:00:01)
- ...

Figure 7: TAL example

allow the use of standard XML tools, GUI or scripting.

Furthermore, the TALES languages provides an abstraction of our model. A minimum understanding of the Advene model is the only requirement since TALES is addressing the model structure rather than its XML representation (which XPath would do). Hence expressions can be customized with relative ease, even by someone not aware of the full TALES syntax. Its similarity with path expressions, used in filesystems and also in URls, is expected to make it seem more familiar to the user.

4.3 Enhanced video based on the ECA paradigm

The current Advene prototype implements STBV views through a view mechanism inspired by the Event-Condition-Action (noted *ECA*) paradigm [Pat99] used most notably in active-databases, but also in other domains like workflow management, publish/subscribe technology or e-mail applications. The dynamic nature of Stream-Time Based Views makes the ECA paradigm well fit.

An Advene view is a set of rules, which defines a specific way of rendering the document. Each rule consists in an event, an (optional) condition and a set of actions. A package can define multiple sets of rules (i.e. multiple views). During the playing of the stream, a number of events are generated. One of the most common is *AnnotationBegin*, which indicates the beginning of an existing annotation has been reached. The ECA-engine is then notified of the occurrence of the event, and checks if any rule matches it, i.e. both its event and its (optional) rule condition match. Each matching rule's action is then executed.

Figure 8 gives an example of a rule, using the scenario described in section 2. Its can be stored in an Advene package by using a representation language such as RuleML.

This language is not meant as a full-fledged multimedia composition lan-

Rule *Display watching tips*:
When the event *AnnotationBegin* occurs,
If the type of the annotation is '*watching-tip*'
Then display the content of the annotation as a caption on the video and pause the player.

Figure 8: An example rule

guage, but is rather aimed at end users. Indeed, they are already familiar with this kind of framework frequently met in e-mail applications to define filters for instance. No coherency checks are applied to the rule: the user has the responsibility to define coherent and non-exclusive rules.

4.4 Prototype status and uses

The prototype currently implements most of the functionality presented in this article, with the exception of the query language that we are investigating. The interaction with the user is done through the main application (for application editing and STBV rendering) and web forms (for UTBV editing and rendering). Epoz[Jab03], a WYSIWYG HTML editor which runs in web browsers, has been integrated in order to ease the editing of views.

An experiment is being carried out with cinema studies teachers, in order to use the prototype in a real context. We are first experimenting the use of Advene as a study tool for preparing courses. A second step will test the use of Advene as a teaching tool. Another experiment has been launched with language teachers who would like to use the prototype in class.

The current main objectives of the project are to improve the robustness and ease-of-use of the prototype, and to design and implement a query language.

5 Related works

Some aspects of our work have been partially covered by a number of projects. We will briefly describe them along three different angles: document models, authoring tools and systems, and document rendering.

5.1 Document models

The first aspect concerns document models. Vane [CLAL97] is one of the first attempts to provide a domain-independant model for AV documents annotations, based on SGML. Since then, technologies and standards have evolved, and the main candidate in the multimedia content description is MPEG-7 [SKP02], formally named *Multimedia Content Description Interface*, which provides a rich set of standardized tools to describe multimedia content. The goal of the MPEG-7 standard is to allow interoperable searching, indexing, filtering and access of audio-visual content, and the Advene implementation aims at interoperating with MPEG-7 as well as possible. MPEG-7 defines a representation of the information associated to an AV stream, as well as a means (through XML Schema) to specify description schemes, which we can relate to Advene's schemas. The design of Advene takes into account the existing work done in

MPEG-7, though we do not natively use all of MPEG-7 as a representation for the sake of simplicity. The fragment model is based upon the similar notion of MediaOccurence found in MPEG-7, and we plan to reuse some MPEG-7 Description Schemes, such as the Classification Scheme to define a thesaurus.

We have formerly developed another audiovisual annotation model called *Annotation Interconnected Strata* [EZPMP00] in which fragments are annotated by keywords organized in a graph. Compared to Advene, *AI-Strata* is more focused on knowledge representation than on the documentary and rendering aspect.

Many video annotation models are dedicated to a single domain. One of the most prolific area is the linguistics domain, where the models range from the simple transcription of video [Tho02] to more elaborated annotation models, such as Anvil [Kip01], able to describe gestures and other deictic features of an AV stream. The ATLAS framework [LFGP02] aims at providing an abstraction over the diversity of linguistic annotations, and offers interoperability through this common framework. It is built upon the Annotation Graph model [BL01], which is a formalism directed towards linguistic annotations. One of the interesting elements of ATLAS is their very flexible definition of regions (the equivalent of *Fragments* in the Advene model), which can accomodate very different types of data (audio or video streams, geometric areas, 3D-areas, ...).

The OPALES project [NN01] allows a group of users to work collaboratively on a video corpus. Its largely reflexive model allows the annotation of any element in the system, including video segments, but also other annotations and users. This makes it an interesting platform, but rather complex and necessiting a huge technical infrastructure (video and annotation servers).

5.2 Authoring tools and systems

Another aspect of video annotation projects is document authoring tools and systems.

Many used video annotation frameworks and tools are dedicated to a specific domain, such as linguistics. In this field, the applications range from simple transcription tools such as Transana [Tho02] or TransTool [Kum02] to more complete applications providing a richer set of annotations [Kip01] and the possibility to define specific annotation structures. Other interesting authoring tools are dedicated to producing multimedia documents but do not include an underlying document model: LimSee2 [Wec03] for instance is a powerful SMIL authoring tool designed to ease the manipulation of time-based scenarios, which requires that the user provides the presented data from an external, non-integrated source.

However, none of them are aimed at DVD interaction. They all necessit the availability of the video through a software player that does not support DVDs, thus exhibiting licence issues. Furthermore, they are aimed at analysing videos to gather raw information about them, and do not deal with the annotation processing for visualisation, as we do with queries and views. We argue that, in the field we are interested in (end-users annotating and building a discourse), the processing and display of annotations is bound to the annotations and the annotation schemas to a large extent.

5.3 Rendering

The third aspect deals with the mechanisms that we use to render our native data into the chosen representation.

Concerning UTBV, the main format that we expect to use is (X)HTML, due to its simplicity and the large availability of tools for editing and visualising such content. But any language featuring a good integration of hypertext data along with ease of generation would be a good candidate.

The dynamic nature of Stream-Time Based Visualisation makes the number of available formats more reduced. Some *ad hoc* solutions have been devised, by extending an existing standard such as HTML [SRM00] or integrating with a video player like Quicktime Player, which allows the integration of hypertext links into quicktime video streams. The more generic and open solution seems to be SMIL [W3C01], which defines a language to write interactive multimedia presentations, one of the goals being the reuse of its features in other XML-based languages such as XHTML or SVG. The MAGpie application [MAG03], a tool for creating closed captions and audio descriptions of videos, is an example of an application using SMIL as a target dynamic language. In Advene, SMIL is used as an export language (TAL templates can generate SMIL documents) with the limitation that, to the best of our knowledge, no multiplatform SMIL player can read DVDs.

6 Conclusion and perspectives

In this paper, we have presented the Advene framework for creating, using and sharing annotations on videos in DVD format. This framework has been designed to provide a general annotation model and corresponding tools, enabling communities of users to share and reuse annotations, annotation schemas and ways to render annotations as hypervideo documents (enriched playback, illustrated discourses, etc.).

The choice of the DVD format is the first originality of our approach, since it addresses a large and widespread corpus of videos. Users owning their own copy of the DVD can share their annotations without having to share copyrighted movies. Another contribution of our approach is that it makes annotation sharing easy by bundling all the useful information in a single XML document: an Advene annotation *package*, around which the whole Advene model is centred. Packages contain not only annotations but also schemas and *views*, either User- or Stream-Time Based (UTBV, STBV), specifying how annotations can be rendered. The view mechanism makes packages both hyperdocuments and hyperdocument generators. To demonstrate the capabilities of the Advene model, a prototype has been developed, based on standard, open and portable technologies (including XML and Python).

A salient feature of digital audiovisual documents is the lack of common uses around them, besides plain visualisation and simple navigation. Annotation is often presented with the goal of indexing, in order to efficiently retrieve poorly structured content. However, we believe that any non-trivial use of videos can produce reusable annotations (active reading, editing, etc.). Most of the uses are still to be invented, which requires versatile and open tools. The Advene framework aims at providing such tools. General purpose annotation schemas and views have to be designed in order to bootstrap annotation practices. Then, more specialized uses will be able to emerge, together with associated annotation tools. As we already mentioned, some scientific communities already have precise practices, which can also be implemented over the Advene framework. This is why we plan to release our framework as an open source projet, so that users may extend it with plugins adapted to their needs.

References

[AP99] Gwendal Auffret and Yannick Prié. Managing Full-indexed Audiovisual Documents: a New Perspective for the Humanities. *Computer and the Humanities, special issue on Digital Images*, 33(4):319–344, 1999.

[BHL99] Tim Bray, Dave Hollander, and Andrew Layman. Namespaces in XML. Recommendation REC-xml-names-19990114, Word Wide Web Consortium, 1999.

[BL01] Steven Bird and Mark Liberman. A formal framework for linguistic annotation. *Speech Communication*, 33:23–60, 2001.

[CG02] Teresa Chambel and Nuno Guimaraes. Context perception in video-based hypermedia spaces. In *Proceedings of the thirteenth conference on Hypertext and hypermedia*, pages 85–94. ACM Press, 2002.

[CLAL97] M. Carrer, L. Ligresti, G. Ahanger, and T.D.C. Little. *Multimedia Tools and Applications 5*, chapter An Annotation Engine for Supporting Video Database Population, pages 233–258. Kluwer Academic Publishers, 1997.

[EZPMP00] Elöd Egyed-Zsigmond, Yannick Prié, Alain Mille, and Jean-Marie Pinon. A graph-based audiovisual document annotation and browing system. In *RIAO'2000*, volume 2, pages 1381–1389, Paris, apr 2000.

[FB96] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. RFC 2046, Internet Engineering Task Force, 1996.

[Jab03] Maik Jablonski. *Epoz, a cross-browser WYSIWYG editor for Zope*, 2003. http://epoz.sourceforge.org/.

[Kip01] Michael Kipp. Anvil - A Generic Annotation Tool for Multimodal Dialogue. In *Proceedings of Eurospeech 2001*, pages 1367–1370, Aaborg, Sept 2001.

[Kum02] Sujai Kumar. *Transtool user guide*. Cognitive Development Laboratory, University of Illinois at Urbana Champaign, 2002.

[LFGP02] Christophe Laprun, Jonathan G. Fiscus, John Garofolo, and Sylvain Pajot. A practical introduction to ATLAS. In *Language Resources and Evaluation Conference (LREC)*, 2002.

[LV98] Craig A. Lindley and Anne-Marie Vercoustre. Virtual Document Models for Intelligent Video Synthesis. In *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA'98)*, pages 661–666, Melbourne, Australia, 1998.

[MAG03] Media Access Generator (MAGpie), 2003. http://ncam.wgbh.org/webaccess/magpie/.

[NN01] Marc Nanard and Jocelyne Nanard. Cumulating and sharing end users knowledge to improve video indexing in a video digital library. In *Proceedings of the first ACM/IEEE-CS joint conference on Digital libraries*, pages 282–289, Virginia, USA, Jun 2001.

[Pat99] Norman W. Paton, editor. *Active Rules in Database Systems*. Springer, 1999.

[SBS96] Nitin Nick Sawhney, David Balcom, and Ian E. Smith. Hyper-Cafe: Narrative and Aesthetic Properties of Hypervideo. In *UK Conference on Hypertext*, pages 1–10, 1996.

[SKP02] José Mara Martnez Sanchez, Rob Koenen, and Fernando Pereira. MPEG-7: The Generic Multimedia Content Description Standard, Part 1. *IEEE Multimedia*, 9(2):78–87, 2002.

[SRM00] L.F.G. Soares, R.F. Rodrigues, and D.C. Muchaluat. Authoring and Formatting Hypermedia Documents in the HyperProp System. *Multimedia Systems Journal,*, 8(2):118–134, 2000.

[Tho02] C. Thorn. Creating New Histories of Learning for Math and Science Instruction: Using NVivo and Transana to manage and study large multimedia datasets. In *Conference on Strategies in Qualitative Research*. Institute of Education, University of London, 2002.

[W3C01] W3C. *Synchronized Multimedia Integration Language (SMIL 2.0)*. W3C, 2001. http://www.w3.org/TR/smil20/.

[Wec03] Daniel Weck. *LimSee2 Tutorial*, 2003. http://wam.inrialpes.fr/software/limsee2/tutorial.html.