Jon Heggland

# OntoLog: Flexible Management of Semantic Video Content Annotations

Doctoral thesis
for the degree of doktor ingeniør

Trondheim, 2005

Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Computer and Information Science

**NTNU**
Norwegian University of Science and Technology
Doctoral thesis
for the degree doktor ingeniør
Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Computer and Information Science



ISBN 82-471-7210-0 (printed vers.)
ISBN 82-471-7209-7 (electronic vers.)
ISSN 1503-8181

Doctoral theses at NTNU, 2005:161

Printed by NTNU-trykk

# Abstract

To encode, query and present the *semantic content* of digital video precisely and flexibly is very useful for many kinds of knowledge work: system analysis and evaluation, documentation and education, to name a few. However, that kind of video management is not a trivial matter. The traditional stratified annotation model has quite poor facilities for specifying the *meaning* – the structure and relationships – of the strata. Because of this, it may also be troublesome to present the annotations to the users in a clear and flexible manner.

This thesis presents *OntoLog*, a system for managing the semantic content of video. It extends the stratified annotation model by *defining the strata as objects and classes in ontologies*, thereby making their semantic meaning more explicit and relating them to each other in a semantic network. The same ontologies are also used to define properties and objects for describing both the strata, individual video intervals and entire videos. This constitutes a very customisable, expressive and precise description model, without sacrificing simplicity and conceptual integrity.

Arranging the annotation strata in a near-hierarchical network with specified semantics (classes, subclasses and instances) also enables reasoning about the annotations during query and browsing. In particular, it enables *visual aggregation of traditional timeline-based strata graphics*. Using this to create compact content visualisations, the OntoLog system is able to present tens of videos on screen at the same time, thus providing *inter-video browsing*. By judiciously disaggregating selected parts of the strata hierarchy, users can focus on relevant strata at their preferred level of detail – *overview-and-zoom* functionality for semantic annotations, in other words.

The OntoLog system has been implemented in the form of six Java applications and web services – together covering annotation editing, browsing, analysis, search, query and presentation with various approaches – built on top of an RDF database founded on SQL. The system has been tested under realistic conditions in several real-world projects, with good results. A novel information gathering interface for OntoLog data, *Savanta*, has been created. This is based on an iterative interaction paradigm featuring inter-video browsing, filtering, navigation and context-sensitive temporal analysis of the annotations. In a comparative usability evaluation, Savanta is shown to outperform more traditional user interfaces for video search/browsing with regard to expressive power, straight-forwardness and user satisfaction.

# Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) in partial fulfilment of the requirements for the doctoral degree *doktor ingeniør*. The work has been performed at the Data Base Systems Group of the Department of Computer and Information Science (IDI) in Trondheim, Norway. The study was funded by Faculty of Information Technology, Mathematics and Electrical Engineering (IME), NTNU.

## Acknowledgements

First of all, I would like to thank my advisor, associate professor Roger Midtstraum, for his tireless efforts through the long years; and for interesting discussions, relevant as well as irrelevant.

Second, I would like to thank my long-time friend, colleague and office-mate Jon Olav Hauglid – without whom this thesis would not even have been started – for feedback, discussions and technical assistance, and essential cooperation on the Savanta project.

Third, dr. med. Hallvard Lærum must be profoundly thanked for his influence on the fundamental design ideas of the OntoLog system, and for his constant feedback on the system during its development.

Thanks to graduate students Jørgen Austvik and Per Håkon Meland, Rune Rystad, Cristoph Stengel, Sigve Litsheim, Erlend Agøy Engum, Pål Jødahl and Eivind Staff for their contributions.

Several people have been involved in the testing and evaluation of the OntoLog system. In addition to Hallvard Lærum, I would especially thank Steinar Line for letting me record his lectures to use as test data, and for his help in the Savanta evaluation. Thanks also to Michael Cyrus, Anne Marit Myrstad, Marianne Holand and Kjetil Nørvåg.

Finally, I would like to thank Christian Mönch and Olav Sandstå for their feedback on the thesis; and my friends and family and the chamber choir A Cappellissimo for helping me keep up the spirits during this work.

# Table of contents

# List of figures

# 1 Introduction

The last decade has seen the cost of digital video drop from the realm of professionals to the levels where it is within reach of almost any interested amateur. Video cameras, both analogue and digital, are steadily getting better and better, and cheaper and cheaper. Stock computers are now powerful enough to easily digitize, edit and compress video. Video editing software is included in the latest operating systems, and more advanced systems can be bought relatively cheaply; some are even available as free- or shareware. Storage costs have dropped to the point where you need hours, not minutes, of high-quality video to fill a standard hard disk, and advances in network capacity are finally making the video-on-demand dream come true.

This development leads naturally to a huge mound of digital video that needs to be managed. There are issues of production, compression, storage, indexing, transfer, protection, dissemination, access and description to be studied. This thesis focuses on one aspect of digital video (and similar media): the management and use of advanced metadata – specifically temporal, semantic annotation describing the content of the video according to the needs and requirements of different users, domains and purposes. This chapter describes the motivation for this work, the research questions the thesis will answer, and the methods with which the answers will be obtained. It also includes an overview of the organization of the thesis.

## 1.1 Motivation

Video (or moving images in general) is a very powerful medium. It has the highest verisimilitude ("likeness to the truth") of all the media, and so is eminently suited for documentation and data gathering. It is also very rich – if a single picture says a thousand words, how many words does a video sequence say? However, the information contained in a video is not easily managed. It requires human interpretation in most cases, and is prone to ambiguity. A computer has a hard time figuring out the semantic content of the video – that is, what it is about, not just what it shows (which is difficult enough), so content-based retrieval of video is for many purposes reliant on metadata. If you ask a video database "Show me the scene in this film where the villain explains his evil plan to the captured hero", the computer will probably have to turn to metadata to accomplish it – it is at present very difficult indeed for a computer to extract such high-level content information automatically from the video data.

Video has another distinguishing feature: its time extent. Much of the meaning of video is imparted through the juxtaposition of elements in the temporal dimension. It takes a certain amount of time to watch a video, and you cannot "skim" it in the same way as an article or a book. This calls for a different way of describing video – an approach that considers the time dimension, and that may be used (among other things) as a substitute for the headings and tables of content found in textual media.

### 1.1.1 Semantic content annotations

Semantic content annotations (within the context of digital video management) are units of information that describe the contents of various parts of the video in machine-understandable terms, so that they can be managed by a computer system. These annotations are useful for many purposes.

**Access.** Searching in raw video (and audio) data is a hard problem, and one that hasn't been successfully solved yet. Feature descriptors like colour distribution for images, dominant frequency for audio and shot boundaries for video are easy to produce; however, semantic annotations describing the meaning of the data cannot (yet) be produced by computers alone, since much of this meaning may be a matter of interpretation, or change depending on the context or point of view of the viewer. Thus, in order to improve content-based access to the data, additional annotations are needed.

**Augmentation.** Most media productions are obviously meant to be self-sufficient; nobody needs detailed annotations to enjoy a music video or a feature film. However, media researchers or film historians, who view the material from a different perspective, would benefit from the ability to augment the data with their personal observations and comments – or from reading the observations and comments of others.

**Context.** It is inherently impossible to experience the entire extent of a temporal data item at once. Large spatial data structures have a similar problem – to discern detail, you have to focus on a small part of the whole. In doing so, it is easy to lose track of the big picture, especially if you are referred to a subpart of a large data structure by a hyperlink or a search result, in which case you have little idea what to expect. Judicious use of annotations to visualise an abstract of the surrounding data helps establishing the context of what you're seeing, and may save time compared to actually watching the surrounding video.

**Overview.** Temporal data types like video are by their very nature time-consuming to view. Browsing a video by fast-forwarding is impractical, since it reduces intelligibility considerably, and still takes significant amounts of time. Visualising temporal annotations by mapping them into the space domain alleviates this problem, making it possible to get an overview of the data without spending so much time.

### 1.1.2 Application areas

The biggest arenas for moving images are of course the TV and the cinema, where content annotations may not be in very high demand. However, the verisimilitude and immediacy of video, along with the low and decreasing cost of digital video systems, has led to it being used in many other contexts, where the goal is not simply passive consumption:

**System analysis and evaluation.** Video is often used by system analysts to record the various procedures of a system, in order to analyse it and discover weaknesses and best practises. Likewise, user interface designers have long been using video to record user evaluations (and anthropologists and behavioural

scientists to record human or animal behaviour), since the richness and verisimilitude of video captures the reactions of the test subjects far better and more objectively than a written account of the test. These applications benefit from the ability to augment the video material with comments, explanations and conclusions, and some mechanism for accessing significant segments of the video directly, instead of having to view it from the start. Structured descriptions lend themselves more readily to computerised analysis than a written transcript or prose summary of an event.

**Documentation, history.** Video is eminently suited for documenting stuff. For instance, the Norwegian Museum of Cultural History uses video to document and preserve old craft techniques. However, the videos are not self-explanatory, so additional commentary is needed to explain what is going on at various stages. Likewise, old films are excellent source material for historians, but they need detailed metadata describing what goes on, where, and when.

**Education.** Video recordings of lectures distributed over the Internet is an interesting application of digital video. Students don't need to be present at campus to follow a course, and can view and review lectures at their convenience. Semantic annotations can be used to augment the material by providing explanations, references, examples and discussions, and they can be used for access, allowing students to efficiently find and review the bits of the syllabus they find most difficult. They can provide context, e.g. which lecture parts and topics build on others, or what topics precede or follow the one the student is watching, and for overview, letting a student see at a glance what topics a particular lecture covers.

**Media research.** Last but not least, researchers studying the moving image as a medium might use semantic annotations to analyse the modes, codes and composition of a film. The "rhythm" of a film (how long its shots or other structural elements typically are), its use of camera movements (pan, tilt, zoom etc.), its use of narration or modes of communication (exposition, observation, interviews and so on) are examples of information that it would be useful to annotate a film with.

### 1.1.3 Discussion

The domains above are both similar and dissimilar in several ways. They deal with quite different aspects of video utilisation, but the purpose of semantic content annotation in all of them is the same: *To represent (some aspect of) the knowledge contained in or related to the video*. It is also important, of course, to represent it in a manageable format, so that it can be stored, transferred, presented, analysed and understood in an efficient manner, preferably by both humans and computers. Another common characteristic is the need for *relationships* between the elements of such a representation, both temporal (this comes before that, this happens at the same time as that) and semantic (this is the opposite of that; this is a subtopic of that).

But the actual information that is to be stored is quite disparate in the different domains. The terms used in the education domain – examples, exercises,

paradigms, terminology according to the subject area – are markedly different from (say) the actors, activities and events of behavioural research. Hence, a great deal of flexibility in the description vocabularies is needed. Genericness – use of broad, non-specific description terms – might be an alternative, but that would lead to a lack of preciseness in the annotations that might not be acceptable. If you want an educational domain annotation system able to reliably procure the video clips pertaining to the explanation of Java "if" conditions in a programming course, you need specific, precise descriptions.

To summarise: Semantic content annotations are useful, but not at all trivial to handle, especially if you want expressiveness, preciseness and flexibility in the same package. Consequently, it is a subject worthy of study.

## 1.2 Research questions and approach

The goal of this thesis is to investigate the issues related to supporting detailed semantic content annotations for the purposes and applications described in the previous section. The main questions it intends to answer are:

1. **Video information modelling:** How should a model for semantic, temporal annotations be constructed, given that it should be usable for quite different domains, purposes and levels of detail?
    - What are the requirements concerning such a model? What kinds of semantics, what level of expressiveness should be supported?
    - How can the model be made flexible or extensible enough to support different kinds of applications, while not sacrificing simplicity and usability?
2. **Tools and interfaces:** How can one create an infrastructure to handle the various tasks related to the usage of semantic content annotations?
    - What kinds of architectures and technologies can or should be used? How should the metadata be stored, accessed and transferred to the user?
    - How should tools for entering, browsing, presenting and querying annotations be constructed?
3. **Usability and performance:** How does such a system fare in practice, in the real world?
    - Does it provide real benefits compared to prior approaches found in literature? Under what circumstances is it appropriate and successful, and when is it not? Is there an actual need for this kind of system?
    - Are the model and the tools user-friendly enough? Expressive enough? Extensible enough? Simple enough?
    - Is the model possible to index and search efficiently? Is it scalable? How does the choice of architecture/technology affect this?

[Denning et al. 1989] defines three paradigms for computer science research: Theory, Abstraction and Design; rooted in mathematics, the experimental scientific method, and engineering, respectively. Theory is, as its name suggests, concerned with developing a coherent, valid theory; Abstraction with investigating a phenomenon, and Design with constructing a system or device to solve a problem. The steps associated with each are presented in the table below.

| | Theory | Abstraction | Design |
|---|---|---|---|
| Step 1 | Characterise object of study (definition) | Form a hypothesis | State requirements |
| Step 2 | Hypothesise possible relationships among them (theorem) | Construct a model and make a pre-diction | State specifications |
| Step 3 | Determine whether the relationships are true (proof) | Design an experiment and collect data | Design and implement the system |
| Step 4 | Interpret results | Analyse results | Test the system |

I consider the Design paradigm to be the best approach in this case, due to the nature of the problem and the results I wish to obtain. There is a practical problem to be solved – the management of complex semantic video content annotations – and a natural way of solving it is to construct a computer system, based on the requirements of users, and to evaluate and test it with end-user satisfaction as the primary goal.

Thus, the first step in addressing the questions posed earlier is to understand the problem. This entails examining the characteristics of video and the requirements of different applications that benefit from semantic content annotations. It also involves studying different standards, technologies and techniques that may be of use in constructing an annotation system, and identifying, examining and evaluating the already existing models and systems.

Building on this foundation, the next step is to suggest an abstract model that fulfils the requirements I have identified as much as possible, and implement this model using a set of carefully chosen technologies. This naturally leads to the construction of tools for the most important tasks – entering, browsing, querying and analysing annotations – and the identification and discussion of the issues related to the design of each of them. The making of such tools and user interfaces is a challenging and interesting research topic in its own right, and it is also a prerequisite for evaluating the model under realistic circumstances.

Finally, the designs I have come up with should be evaluated, according to several criteria – expressiveness, flexibility, user-friendliness, performance and scalability, to mention a few. I will thus reach a conclusion as to whether my designs solve some or all of the problems and challenges related to semantic video annotations, and identify which areas and issues should be studied more.

The design steps are not performed in isolation, nor necessarily in a strict order; it is expected that iterations will occur, as knowledge produced by later steps leads to a re-examination of earlier steps. In this case, it is likely that the development of tools and interfaces will influence the design of the model, and vice versa; that early evaluations of the system will lead to redesign and further evaluations; and that realistic tests with actual users will lead to a greater understanding of their requirements.

My approach also contains elements of the Abstraction paradigm, in the design of the model as well as in the user interface design and evaluation. This is to be expected; Denning et al. [ibid.] points out that in computer science, the three paradigms are often intricately intertwined.

The plan for the design is to start with a set of high-level scenarios, to explore the problem space and establish preliminary requirements. These scenarios will be revisited later, to validate the design, but the bulk of the process will be user-centred [Gould and Lewis 1985]: iterative design of conceptual models and prototypes, with input from actual users adjusting the development. In any case, this thesis will focus on the results rather than the process.

## 1.3 Contributions

The major contributions of this thesis are:

*A flexible, expressive, structured and simple model for semantic annotation of temporal media.* I consider existing approaches lacking in one or more of these aspects, so I propose a new model based on augmenting traditional stratified annotations with ontologies. The model is geared towards the purposes and domains outlined in section 1.1, and has several interesting properties: It allows users to precisely define their universe of discourse, and alleviates the problem of *topical granularity* (at what level of detail should a video be described?).

*Demonstrations of this model's viability in several domains.* An implementation of this model, the OntoLog system, has proved its worth in several real-world projects from quite different disciplines: medical informatics, education and media research.

*A powerful visualisation scheme for stratified temporal annotations in video databases, enabling better overviews and inter-video browsing.* The annotation strata are organised hierarchically in an ontology, with subset (subclass) or member (instance) semantics. This makes it possible to aggregate the visualisation of related strata, making the representation more compact and hiding unnecessary details. The compact representation gives room for visualising and browsing several videos at once (inter-video browsing), and the users may easily disaggregate or "zoom in on" the strata they are particularly interested in.

*A novel and powerful information gathering interface, based on integrated browsing, navigation, visualisation, searching and filtering, with demonstrated benefits.* The application named Savanta provides a rich environment for video browsing and analysis, utilising the richness of the model, the powerful visualisation scheme and context-sensitive temporal analysis of the annotations.

Its flexible and iterative interaction model provides a user-friendly approach to complex information gathering needs.

*A demonstration that RDF is a viable alternative for temporal video annotation.* The Resource Description Framework [World Wide Web Consortium 2004a] and related technologies were used for the implementation of the OntoLog model. Though primarily aimed at describing web pages and other kinds of documents, RDF's flexibility, expressiveness and simplicity has been shown to be well suited for video metadata as well.

## 1.4 Thesis outline

This thesis is organised into three main parts. The first part deals with the background of this work. Chapter 2 discusses the characteristics of video, its properties as a digital data type and the complexity of its semantics. Chapter 3 examines the state of the art in video content models, tools and user interfaces, evaluates their weak and strong points and discusses where to go next.

The second part deals with the design and implementation of the OntoLog system. Chapter 4 introduces a set of scenarios as examples of the various uses for semantic annotations, presents a set of requirements for an ideal multi-purpose model, discusses the main ideas behind the OntoLog approach, and gives an overview of technologies and standard relevant to this. Chapter 5 presents the OntoLog model for temporal semantic content annotation, and chapter 6 describes how the novel features of this model can be harnessed by tools and user interfaces that give the model life – tools for constructing, presenting, browsing, analysing and querying annotations.

The final part consists of evaluations of various aspects of OntoLog, and discusses what conclusions and reusable knowledge may be gleaned from this work. Chapter 7 discusses various cases where the OntoLog system has been put to the test in real environments, and examines how it fares in each case; it also presents a quantitative, comparative evaluation of the information gathering interface Savanta. Chapter 8 summarises and discusses the successes and limitations of this project, and chapter 9 concludes the thesis.

# 2 The video medium

Video or film is quite different from text, with regard to both how it is stored and managed digitally, and how it is "read" or understood by the viewer. For this reason, a system for describing, indexing, querying and presenting video must necessarily be different from a corresponding system for textual information. In this chapter, I explore the characteristics of video in this regard. In section 2.1, I briefly mention its properties as a digital data type. Section 2.2 discusses the semantics of video: the factors that determine how it is perceived, interpreted and understood as a means of communication. Section 0 discusses the discrepancy between the digital representation of video and its semantics, and section 2.4 summarises the chapter.

## 2.1 Video as a digital data type

Video/film is in essence a series of equal-sized images (called frames), displayed sequentially at a fixed rate (typically 25 or 30 frames per second) to create the illusion of movement, together with zero or more synchronised audio tracks. The frames are encoded digitally as bitmap images, and compressed with various techniques. A common compression technique is based on the idea of storing only the differences from one frame to the next, since two consecutive frames are often very similar. The audio tracks are usually also compressed; however, it is not very important for this thesis exactly how video digitalisation and compression is performed.

Despite compression, video data is still very voluminous. A VHS quality motion picture of standard length, compressed using standard techniques and algorithms, typically range from 650 MB to 1.3 GB, since it is desirable to fit a films on one or two standard CD-ROMs. A DVD movie with its noticeably higher image and sound quality may be more than five times that size [Chambers 2002]. This is a challenge for digital video management and handling. One thing is the amount of storage needed for archives and caches. Another is the infrastructure needed for transport; network bandwidth is a bottleneck compared to storage capacity and processing power. This is a big issue for video archives where the video files are not stored locally on the user's computer.

There are a few options available when delivering video data to the end user across a network. One is to download the entire video file, but this is most likely very time consuming. This can be alleviated by starting playback of the video before it has been entirely downloaded, but the user is then constrained to watch it from the start, and may have to wait for more data if playback catches up with download. A final option is *streaming*, where the video data is delivered to the user as and when it is needed. This reduces the storage requirement of the client, since less data has to be cached, and may offer the end user random access to the video material: If s/he skips to a different part of the video, streaming stops and is restarted from the desired time point. However, this technique places very high demands on the quality of service (QoS) of the network link; it has to guarantee extremely timely and reliable data transport, which is difficult to do over the Internet. I will not go into details about storage and network technologies either,

but the challenges introduced in this section will of course be relevant for the goals presented in chapter 1.

Another video management challenge related to the common digital representation of video is the *semantic gap*, which will be discussed later in the following section.

## 2.2 Video semantics

Video has an immediacy of expression that is quite unique. Written and spoken language needs to be read, interpreted – the sequence of letters or sounds comprising the word "bird" is very different indeed from an actual bird. In contrast, a video image of a bird is considerably more similar to the real thing. Very little background knowledge or learning is required to make the connection between the representation and the actual bird (the *signifier* and the *signified* in the terminology of semiology, the study of signs); an infant, or even a cat, can do it.

This might lead one to believe that "reading" a video is very simple: What you see is what you get. This is a misleading impression, as will be explained in this section. For one thing, video can incorporate other modes of expression, since it can record them: spoken language and music can be included in the soundtrack, written language as titles; dance, painting and other visual arts can be shown. Other issues include the physiology and psychology of seeing an image; different kinds of signs (in addition to the simple sign exemplified by the bird example above); and above all the decisions of the filmmaker on what to show, how to show it, and how to present it. These are all important factors in the semantics of video; an "illiterate" video viewer may not notice them consciously, but is nevertheless affected by them. As film semiologist Christian Metz says, "A film is difficult to analyze because it is easy to understand." [Metz 1974]

### 2.2.1 "Reading" an image

Two people watching the same image may experience the same optical pattern, but that does not mean they see the same thing. Physiological and cultural phenomena affect the mental image each viewer constructs to a significant degree.

We read a page of text in a fairly strict and predictable pattern – sequentially, from top to bottom and left to right, in the case of this book. We do not read an image in the same way, but neither do we view the whole image at once: The physical construction of our eyes allows us only to see a remarkably tiny area clearly, so we must continuously move the eye to perceive a bigger object in detail [Findlay and Gilchrist 2003]. These half-conscious movements (called *saccades*) tend to follow certain patterns in the image, but can be more or less extensive and efficient, determining attention to detail and speed of apprehension.

Cultural differences also affect image understanding. Westerners, trained in left-to-right reading, tend to think of objects to the left in a picture as coming "before" objects to the right, and consider diagonals from bottom left to top right as

“ascending”. People with right-to-left or top-to-bottom reading patterns interpret such relations and patterns differently. It is a question of background and learning; what you get is not determined solely by what you see.

Another example is perspective drawing, often used in optical illusions of various kinds. Consider the drawing in Figure 2.1: That this represents a cube is obvious only to a reader trained in western conventions for perspective. It has been shown [Deregowski 1972] that different cultures interpret such images very differently. For that matter, simply rotating the drawing by 45° makes it far less likely to be interpreted as a cube, since by convention the depth dimension is indicated by oblique lines.

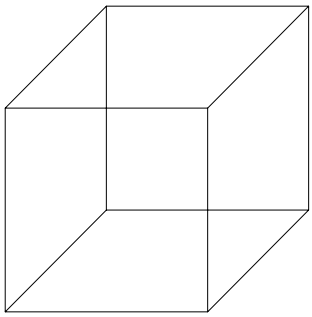

**Figure 2.1: The Necker cube**

Figure 2.1 also illustrates another psychological phenomenon: the multistable image, an image that can be interpreted by the brain in several equally valid ways. Is the cube seen from the top or the bottom? Both, but not at the same time, and switching between interpretations may be conscious or unconscious, effortless or difficult, depending on the viewer.

### 2.2.2 Signs and connotation

As Monaco [Monaco 1981] is fond of saying, film is not a language, but it is *like* a language. It has no grammar, and cannot be scientifically and objectively analysed. Nevertheless, it has many characteristics of language, and can be studied successfully using some of the same tools. One of these tools is *semiology*, the study of systems of signs.

Wollen [Wollen 1972], drawing on the philosophy of C.S. Peirce, identifies three kinds of signs in video: Icon, Symbol and Index.

**Icons** are signs where the signifier more or less equals the signified: an image of a bird indicates the bird itself. This is the characteristic sign of video mentioned in the beginning of this section, and one of its great strengths: it is extremely efficient at representing physical reality, and the viewer needs no training for this kind of sign. However, representing more abstract notions is another matter.

**Symbols** are signs where the signifier is connected to the signified solely by convention. Written and spoken language consists of symbols; the word “bird” is not a bird except by mutual agreement. In video, an eagle might signify the United States of America, even though there is no natural relationship between

them (other than that an eagle is a powerful raptor with a rather small brain [Galtung 2002]). Symbols, unlike Icons, require shared knowledge between the filmmaker and the viewer, as well as active interpretation on the part of the viewer.

**Indexes** are something between the Icon and the Symbol. The signifier is not identical or equal to the signified, but neither is it wholly different: It has an inherent, natural relationship to the signified. For instance, clocks are Indexes of time; skulls and tombstones of death; marching feet indicate war. Indexes are excellent means for videos to represent abstract ideas, which otherwise would be very difficult.

The categories are not mutually exclusive. All cinematic signs are Icons, though they may be Symbols or Indexes as well. A Symbol may be more or less ambiguous; the eagle might represent czarist Russia, or be an Index of flight or hunting, while the letters "USA" leave no doubt as to what they stand for. Icons are mainly *denotative* – they are what they are – while Symbols and especially Indexes have a significant *connotative* aspect: they can mean different things, separate from what they themselves are, and their interpretation is affected by several things – the mindset, cultural background and knowledge of the viewer, as well as how they are presented by the filmmaker: the context and presentation, which is the topic of the next two subsections.

### 2.2.3 Mise en scène

According to Monaco [Monaco 1981], three questions confront the filmmaker: what to shoot, how to shoot it, and how to present the shot[1]. *Mise en scène* is concerned with the two first questions, *montage* (cutting or editing in American terminology) with the third.

Mise en scène is often referred to as the spatial composition of a shot, but this is a little imprecise. It concerns not only the placement of entities on the screen, but a host of other cinematic codes the filmmaker has at his disposal: lighting (amount, colour, direction, contrast); camera distance (close-ups, long shots), angle (low or high, oblique or straight), focus (deep or shallow, soft or sharp) and movement; colours (warm, cold, dull, vibrant, strong, washed-out); point of view (first-person, omniscient); framing (open or closed), location/situation and so on.

All these choices affect the "reading" of the shot, how a viewer experiences it – the *paradigmatic* connotations, as Metz [Metz 1974] calls it. The viewer compares the shot, consciously or unconsciously, to other *potential* shots of the same subject. Camera angle can make a subject seem looming and overpowering, or small and insignificant; lighting can make it seem lugubrious, furtive or radiant – all the choices have the potential to change or affect the meaning, mood and importance of (the various elements of) a shot considerably.

---

[1] A *shot* is "a single piece of film, however long or short, without cuts, exposed continuously." [Monaco 1981]

Mise en scène is sometimes presented as static – like the arrangement of still photographs – but this is not strictly true. Many of the codes mentioned above have dynamic elements. Camera movement (pan, zoom, tilt, dolly, track) and movement of actors and objects are obvious examples, but also elements such as focus and lighting can be changed during a shot with significant effect. Nevertheless, mise en scène may be thought of as spatial and static compared to its temporal and dynamic companion, *montage*.

### 2.2.4 Montage

Montage is the seemingly simple process of splicing together shots to create a bigger narrative. This is also a powerful tool for the filmmaker to affect the interpretation of his shots; indeed, it is possible (and common) to use the juxtaposition of two shots to create a third meaning out of the "original" two meanings of the shots. An experiment performed by Russian filmmaker Lev Kuleshov illustrates this: he intercut three identical shots of an actor with shots of a dead woman, a plate of soup and a little girl; and viewers marvelled at the actor's subtle ability to convey such varied emotions as grief, hunger and affection [Monaco 1981].

Clearly, the connotation of a shot depends greatly on the comparison with the shots surrounding it. This is the *syntagmatic* connotation, in Metz's terminology [Metz 1974]. Montage might be called the grammar of video – it is in some ways comparable to constructing statements, sentences and paragraphs in written language – but the analogy is not perfect. A shot is not a word (or a sentence); it is more complex, diverse and ambiguous. There are no grammar rules in filmmaking; thus, a filmmaker has far more power, options and influence when constructing his "sentences" than a writer has. Hence, the syntagmatic aspect is often considered more "cinematic" than the paradigmatic aspect of video, which is more comparable to other arts.

While there are not any rules, there certainly are conventions and common patterns of montage. Metz has suggested a classification of different types of montage, or *syntagmas*, organised in a hierarchy shown (slightly simplified) in Figure 2.2.

At the top level is the autonomous segment, whose meaning is not dependent on the segments preceding or following it. This can either be a single shot, or a syntagma composed of several shots. Syntagmas can be achronological – using flashbacks or flash-forward, alternating between chronologically unrelated stories – or chronological. A chronological syntagma can be descriptive – e.g. establishing a location or situation – or narrative, telling a story. A narrative syntagma can be alternate, showing elements that are parallel but related, like the pursuer and pursued in a chase scene; or linear. A linear narrative syntagma can either have a continuous succession of events, in which case it is a *scene*; or it can be broken up (while still being chronological, linear and narrative) which defines it as a *sequence*.

Montage is used for a myriad of purposes: to create continuity between events in the video; to bend the timeline, provide digressions and forecasts; to set or alter

the tempo and rhythm of the narrative; to compare or contrast elements and stories; to simply avoid "dead time" by removing uninteresting footage, to mention a few. Additionally, the boundaries between shots may also be used to special effect: Different kinds of transitions, wipes and fades affect the interpretation of a cut; drawing attention to the change, contrasting the connected shots or linking them unobtrusively.

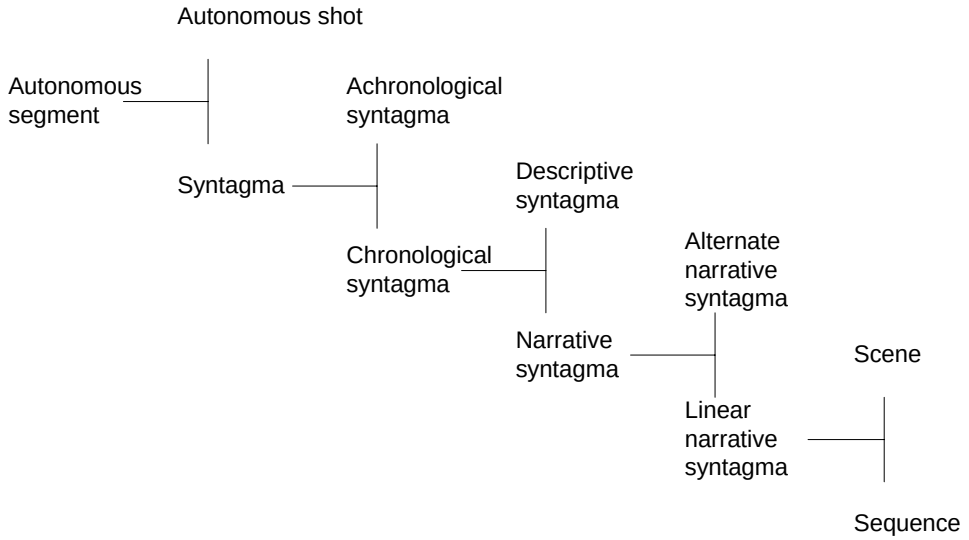


**Figure 2.2: Metz's syntagmas (slightly simplified)**

## 2.3 The semantic gap

As is evident by the discussion above, there is a huge discrepancy between the way video data is coded digitally, and the way it is experienced by a human user. As far as a computer is concerned, video data are sequences of bitmap images – grids of coloured dots – along with a time-dependent air pressure function, i.e. the audio track. The user, however, doesn't see a bunch of dots constantly changing colour – s/he perceives people, buildings, vehicles, landscapes, places, actions, events, discussions, stories, ideas. Most likely, s/he wants to interact with a digital video library using such high-level concepts, not using the vocabulary of coloured dots. The audio data present the same problem: The user has a high-level perception of the voices of different people speaking, their roles and moods and what they are discussing; speeches, cries, fights, music, noise – while the computer considers everything a homogenous sequence of air pressure measurements, taken several thousand times per second. This difference of perspective between the computer and the user is called the semantic gap.

### 2.3.1 Representing and understanding information

This semantic gap is an issue with all kinds of information representation, but video has it worse than most other mediums. For instance, a diagram of boxes of various shapes connected by lines and arrows is actually stored by the computer

as shapes and connectors. The computer understands the diagram to a certain degree, and can determine which shapes are connected directly or indirectly, and what happens if a particular shape is moved. Likewise, a textual document is represented using letters, words, sentences, paragraphs, sections and headings, and the computer can answer simple queries about the actual meaning of the content.

Not so for video. No matter what the message is, it is encoded using the coloured dots. This is of course powerful and versatile – video can represent anything that can be visualised; it is not limited by its vocabulary the way a diagram representation scheme or textual document format is. The downside is that the computer can offer little help in interpreting and understanding the contents. It cannot answer even simple questions like whether a particular person is shown somewhere in the video without having this information explicitly added to the system by an image analysing tool or a human.

Additionally, the users are left alone with the task of constructing information from the video data they are shown, and this process is not trivial. The correct interpretation of Indexes and Symbols requires knowledge and familiarity. Many cinematic codes are culturally dependent. The viewers might need background information on (or just plain identification of) the people, places or events they are watching, and this is simply not present in the video unless explicitly put there by the video author – which it sometimes is, but not if the video is used outside its originally intended context (like when a tourist video accidentally documenting a crime is used as evidence in a criminal case) or audience. By its very nature, video is rich and ambiguous; it records more than just the intention of the video maker. For instance, footage of a soccer match could be used to study crowd behaviour or supporter fashion by watching the spectators instead of the game. The video would probably contain explicit information about who the different players are, and what is happening (goals, penalties, substitutions and such), but probably nothing about the spectators – except the coloured dots.

### 2.3.2 Shot detection and segmentation

There is one cinematic code that is reasonably computer readable, though: the shot boundary. Shot boundaries (or *cuts*) can be detected by looking for abrupt changes in the video image from one frame to the next, a process called shot detection. However, this handles only one tiny aspect of the semantics of video. As shown in the previous section, a shot is seldom autonomous; its meaning is affected by the other shots in its syntagma. Shot detection tells us nothing more than the length of the shot, and thus the rhythm of the video at that point; while this may be useful (the rapid cutting of the famous shower scene in Hitchcock's *Psycho* may indicate that something dramatic, perhaps violent, is happening), it is not sufficient for a proper understanding of its content.

Analogous to shot detection, audio analysis can be used to partition the video into segments. In newscasts, a jingle may signify the transition from foreign news to sports. A speaker uses pauses and changes in emphasis and inflection to announce the beginning of new topics or sections of his lecture; in an interview, the changes

of speaker corresponds to the exchange of questions and responses. This can be recognised by a computer. But again, this tells little about the *meaning* of the video. It is more akin to parsing a body of text into words or phrases (though this analogy should not be carried too far); it is a useful aid and starting point for reading a film – a causeway for bridging the semantic gap – but in most cases, human effort is required to complete the crossing.

## 2.4 Summary

Video is a complex medium. It has an unsurpassed ability to concisely describe physical reality, but its semantics can be relatively obscure: It is extensively based on signs whose meaning are given by relationships – natural and intrinsic or arbitrary and conventional – that must be known or intuitively understood by the viewer. It is affected by complex cultural, psychological and physiological phenomena. We read and understand video by comparing what we see with what we do *not* see, or have just seen (or even will see later).

Digital video is unwieldy due to its size, and its representation bears little resemblance to the ways humans understand it. Because of the importance of syntagmatic connotations, it is time-consuming to watch; fast-forwarding, skimming and skipping harms or destroys the temporal integrity of the video, and thus makes it harder or even impossible to understand properly. This is true for the paradigmatic connotations as well, since they also can have dynamic, temporal aspects. In contrast to the letters and words of written language, the signs and codes of video are not easily alphabetised and indexed; and their meaning is also significantly more context-dependant and mutable. These are all issues that a computer system for handling the semantic content of video must take into account.

# 3 A survey of semantic video management

This chapter describes the state of the art in semantic video management systems – systems that focus on describing, indexing and augmenting the content of video with machine-readable semantic annotations. The primary aim of this survey is to evaluate and compare the expressiveness and flexibility of different semantic video content models found in literature, but also to present paradigms of interaction with actual systems based on such models – tools and user interfaces, in other words.

Section 3.1 introduces four dimensions to describe such models along, while section 3.2 presents a representative selection of the state of the art. User interfaces and tools are mentioned in some of these cases, but such information is incomplete, sporadic and hard to come by, compared to information about the conceptual models themselves. For this reason, user interfaces and tools for video database systems are presented separately in section 3.3. Section 3.4 summarises the chapter.

## 3.1 Properties of video content models

To be able to discuss video content models and their properties, we need a set of dimensions to place them along – some metrics of their capabilities, their expressiveness and complexity.

The most significant property of a video content model is perhaps what features it provides for subdividing the video. Structurally, digital video can be considered a three-dimensional data type: the video image has two dimensions, and the temporal extent provides a third. The video can thus be subdivided both temporally and spatially, and this can be done with varying expressiveness and complexity. These properties I call the *temporal* and *spatial expressiveness*, presented in subsections 3.1.1 and 3.1.2 respectively. The audio track is also in a sense a subpart of video, and the ability to describe it independently and separately from the image part may be desirable. However, I consider this a property of temporal expressiveness, and discuss it further in the relevant subsection.

Given a set of video (and/or audio) fragments created by temporal and spatial subdivision, the next question is how to annotate them. Description schemes can be more or less complex, depending on how much structure, expressiveness and formality is required or desired. This I call the *semantic expressiveness* of the model, discussed in subsection 3.1.3.

Another important measure of a model's worth is how adaptable and extensible it is. Users benefit from the capability to tailor the model to their specific domain and purpose, to customise it to represent particular aspects of the video content at a particular level of detail. Of course, this must be done with care, so that the adapted model still is compatible with the tools and systems supporting it. This is the *flexibility* of the model, presented in subsection 3.1.4.

### 3.1.1 Temporal expressiveness

If one of the defining properties of video is its temporal extent, then the treatment of time may be the most important part of a video content model. Videos may deal with many different topics at different times, and it is important for a content model to be able to reflect this. This is done by marking descriptions as valid only in certain temporal intervals – or, from the other point of view, divide the video into temporal fragments (logically, not necessarily physically) and attach descriptions to the fragments. A content model may put restrictions on these fragments – where their start- and endpoints may be, their size, whether overlapping is allowed, whether adjacency is required, and so on. This dimension I call the *temporal expressiveness* of the model. The most significant points along this dimension, from most to least restrictive, are illustrated in Figure 3.1 and discussed below.

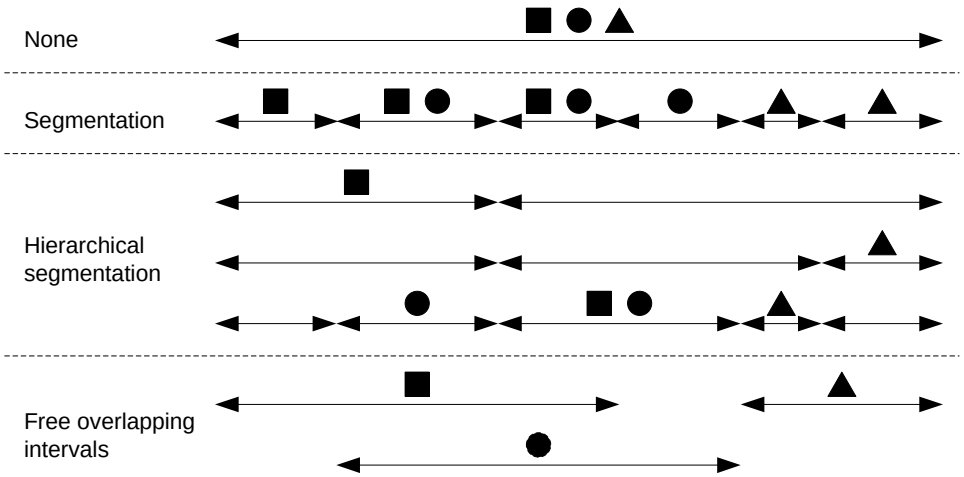


**Figure 3.1: Temporal expressiveness**

#### None

Some models have no representation of time at all; the video is treated as an atomic object. The Internet Movie Database, IMDB[1], is an example of this (though it doesn't actually store the movies, just descriptions of them). Movies are described as a whole, with actors, directors, producers and so on – data that are valid for the entire film.

The advantage is simplicity: The conceptual model ignores the temporal dimension, and treats videos just like books or documents in a conventional library. The obvious disadvantage is that time-dependent descriptions either cannot be entered, or must be "faked". The videos can for instance be physically divided into smaller pieces, and each piece described individually. However, the pieces will in that case all have mostly identical non-temporal descriptions, leading to duplication of data and redundancy problems. Additionally, it is non-

[1] http://www.imdb.com/

trivial to determine the optimal division points, and extremely cumbersome to change them at a later time. Another option is to use time codes entered into unconstrained description fields, but due to their lack of enforced structure, such descriptions are relatively useless for the data management system, and therefore impractical for humans as well.

### Segmentation

With the term "segmentation", I mean logically dividing the video into adjacent, non-overlapping parts – a partition, in other words. Non-temporal (or content-independent) information is connected to the video as a whole, while temporal descriptions are connected to the relevant segment(s).

There are several paradigms for determining segment boundaries. *Fixed segmentation* entails that each segment has a fixed, content-independent size. The advantage of this approach is that it is simple – only the small, simple segment number needs to be stored to be able to compute the corresponding time interval, and the creation of segments requires no human interaction. The downside is that what happens in the video will probably not correspond very well to the arbitrary segmentation. Descriptions will be only partially valid if a segment contains changes in subject, and may need to be duplicated across segments if an interesting sequence straddles a segment boundary. As a simple example, consider annotating a video by marking the presence of different people in the picture: If a certain person is present in several consecutive segments, this has to be recorded separately for each segment; and if a person leaves the picture in the middle of a segment, the annotation is only partially correct.

*Structure-based segmentation* is a more common segmentation scheme: Using physical, structural properties of the video to determine segmentation boundaries. Shot detection (as mentioned in section 2.3.2) is an excellent candidate for this; audio analysis is another possibility. This scheme is only marginally more complex than fixed segmentation, and has the advantage that the shots are often correlated to the semantic structure of the video. However, if the video (or significant parts of it) has no shots, or very many shots, this scheme has the same problems with redundant or partially invalid descriptions.

With *user-defined segmentation*, the user is free to choose the segmentation boundaries that are most useful to her, based on how s/he wishes to describe the video – s/he can tune the segmentation to correspond exactly to the semantic content s/he wants to model. The structure-based segmentation may be provided as a starting point, but segment boundaries can be set arbitrarily (as long as the subdivision still is a partition).

This scheme alleviates the problem of redundant and partially invalid descriptions, since the user has the freedom to add, move or remove segment boundaries, but it does not remove it completely. Some kind of content annotations, e.g. the presence of people, overlap naturally, while segments do not. To model a video sequence where person A enters a room, person B enters later, and then leaves again, one has to use three segments: first a segment where A is alone, then a segment where both A and B are present, and finally another

segment with just A. Conceptually, this is just two pieces of information, not three: A is present in a certain interval, and B is present in another interval.

A general problem with segmentation is that it is necessarily keyed to a specific description approach, and unsuited for describing more than one independent aspect of video content. For instance, music is often used for transitions between scenes in movies, so shot-based and audio-based segmentation would conflict. An existing segmentation may not be easily reused if someone wants to describe the video further from a different point of view – it may be too coarse, too fine-grained, or have boundaries at inappropriate places.

### Hierarchical segmentation

A common method for increasing the expressiveness of the segmented approach is to have several layers of segments, arranged in a hierarchy. Typically, the lowest level is the shot segmentation mentioned above, while the layer above group consecutive shots into scenes (loosely defined in this context as a group of shots depicting some common time, place and/or event); and perhaps a top layer grouping scenes into *acts* or *sequences* (again a more loose definition of the word, compared to Metz's syntagmas in section 2.2.4).

This is a significantly more complex way of doing things than the simple segmentation schemes, since this additional layer dimension has to be handled. To find the descriptions that are valid in a given instant, the system has to consult all the layers in the model, not just one. It is also difficult to automate the construction of the higher levels, due to the semantic gap, and some videos (e.g. unedited surveillance tapes) may not even have any inherent hierarchical structure, rendering this scheme rather inappropriate and impractical. The two- or three-level hierarchy is a quite coarse simplification of the syntagmatic structure of film discussed in section 2.2.4, and may in many cases be too restrictive.

On the other hand, with strictly and regularly structured video material, for instance television news, the hierarchical scheme lets the user select the most appropriate level in which to enter a description, thus avoiding problems due to the segmentation being to coarse or fine-grained. The scene level is by definition homogeneous with respect to many modelling aspects (time and place, usually), which makes annotation easier and more efficient.

### Free overlapping intervals

The fullest expressiveness and flexibility is obtained by removing the constraint that intervals must be adjacent and non-overlapping, thus allowing them to be independent of each other. This makes it possible to annotate independent aspects of the video accurately and precisely – the problems of redundancy and partially invalid descriptions are removed. The video can be described from several different viewpoints and with varying level of detail without interference, as there are no "global" decisions on the placement and size of the intervals. The audio may be described independently of the image content.

This power comes at the price of complexity, however. To find all the descriptions valid at a given point in the video, potentially all the intervals have to

be considered – unlike the segmented schemes where at any time just one single interval (or a fixed number of intervals, in the hierarchical scheme) is known to be valid. The start- and endpoint of each interval have to be stored, compared to just the positions of boundaries between segments. Last but not least, it is more difficult to build good user interfaces for creating and viewing such collections of intervals. Segments are inherently ordered and well-structured; they may easily be represented and visualised using a simple list, a timeline or something similar. Free overlapping intervals have no natural organisation.

A common technique for alleviating this organisational problem is to create an organisation of the intervals by arranging them in layers called *strata* [Chua et al. 2002], each stratum corresponding to some entity or concept – a person, a topic or a place, for instance – in the universe of discourse. This requires some a priori notion of what is to be described in the video, but that is not necessarily a bad thing. It increases the complexity of the model somewhat, as these entities or concepts have to be managed as well as the intervals, but that may be a small price to pay for the quite intuitive structure they impose on the otherwise disorganised intervals.

However, though the free overlapping intervals scheme has a very fine temporal granularity, this stratified approach imposes another kind of granularity: How does one choose what level of detail to use when creating strata? How does one organise them, and determine their boundaries, how they partition the universe of discourse? That, though, is not a temporal problem, and thus not discussed further here.

### 3.1.2 Spatial expressiveness

In image databases, it is not uncommon to be able to subdivide the images spatially, or define regions that can be annotated. The same can be done for video, since video can be thought of as a series of still images; it is also potentially useful, since a lot of information is conveyed through the spatial juxtaposition of elements in the video image, as described in section 2.2. This kind of subdivision is in many ways analogous to creating temporal video fragments – the point is to designate a continuous part of the video for the purposes of annotation. The precision with which spatial, possibly time-dependent regions may be specified is the *spatial expressiveness* of the model. The most significant points along the dimension of spatial expressiveness are:

**None**

It is quite possible to ignore the spatial dimension altogether. In that case, the video fragments are wholly temporal, and descriptions are regarded as concerning the whole video image, or some self-evident part of it. Humans are quite adept at identifying objects in images, and the visual content of a video at any given time is often relatively simple, compared to for instance satellite imagery, maps and aerial photographs.

The advantage of ignoring the spatial dimension is that the video fragments become one-dimensional objects instead of three-dimensional, which saves a lot

of complexity with regard to both computation, storage and user interfaces. The disadvantage is equally obvious: it may be ambiguous what a description refers to. For instance, if the model describes (among other things) the people shown in the video, and two fragments overlap (temporally), the model cannot explicitly express which person in the image each of the fragment descriptions refers to.

### Static regions

Static regions are spatio-temporal regions that do not change with time (except that they begin at some time, and end at some time, of course) – their cross-section along the time axis is always the same 2D shape at the same location. These are relatively easy to handle, as they are not full-fledged three-dimensional objects; their third dimension, time, is severely constrained. The two-dimensional, spatial component may have different levels of complexity or expressiveness – it may be as simple as a circle, ellipse or rectangle, or a polygon or curved outline, or a spatial decomposition scheme such as a quad-tree. A rectangular static region is illustrated in Figure 3.2. The treatment of such data structures is a part of the science of computer graphics, and I will not go into detail about them here. The interested reader is referred to e.g. [Foley et al. 1990].

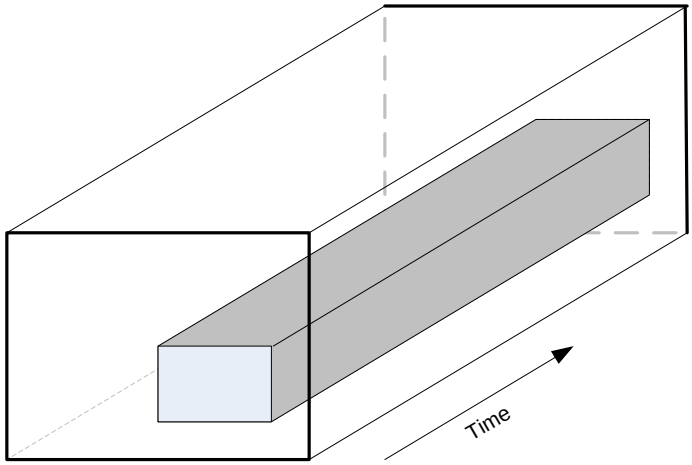


**Figure 3.2: Rectangular static region**

Static regions are relatively simple to handle – each temporal region has an associated 2D geometric shape – but they have the disadvantage that they cannot adequately model moving objects. One would either have to make the spatial region so big that the objects do not move outside it, or use several temporally consecutive regions with slightly different spatial regions. In the first case, the region is imprecise, and may cause ambiguity; in the second case, a conceptually cohesive region is split up, leading to repetitive, redundant descriptions – analogous to the segmentation problems discussed in section 3.1.1.

### Dynamic regions

Dynamic regions are spatio-temporal regions that change with time. Within this class of regions, there are different levels of expressiveness and complexity as

well. A simple kind of dynamic region could be created by specifying a 2D shape and its position at the beginning and the end of the temporal interval, and perform a linear interpolation for the shape's position in the frames in between. This scheme can be extended by allowing more "waypoints" along the temporal dimension, or allowing scaling or other transformations to take place. Figure 3.3 shows an example of such a region.

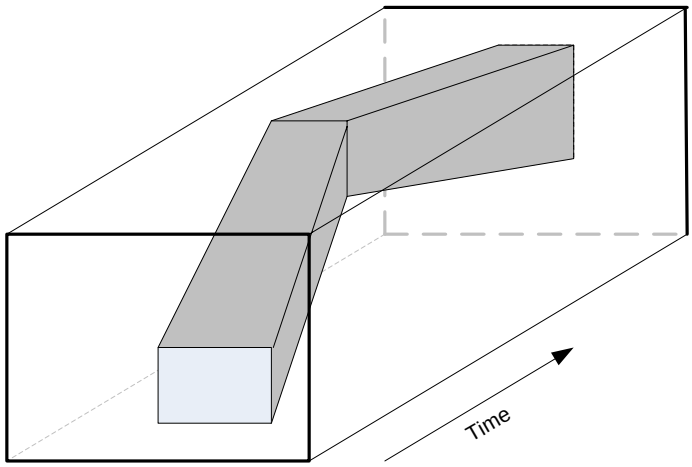


**Figure 3.3: Linearly interpolated dynamic region**

At the far end of the scale, temporal regions are just like any other three-dimensional objects, except that their cross section along the time axis always should be a connected region – that is, it should not split up as the user it watching the video. Again, the science of computer graphics deals with the issues of representing 3D objects, and I will not go into detail here. Suffice to say that the more expressive, precise and general a dynamic region scheme is, the more complex it is to handle.

### 3.1.3 Semantic expressiveness

Given a set of video fragments, the next natural question is how to describe them. What is the structure of the descriptions? What kinds of properties and data values are supported? A content model may support only a single, unstructured description field, or it may provide an advanced type system. This may be called the *semantic expressiveness* of the content model. This dimension has less well-defined and discrete steps than the two previous ones, but the most significant points may be defined as follows:

#### Single, free-text descriptor

The simplest conceivable mechanism for describing a temporal interval is the single, free-text description field. It puts no constraints on the user, so is very flexible – it can be used for anything, in any domain. On the other hand, searching and analysis – trying to find similar descriptions, for instance – becomes imprecise, as computers cannot interpret and index such data very accurately. Large, unstructured descriptions are cumbersome to work with, if one is primarily

interested in only one aspect of them – imagine having to browse through several pages of descriptions just to find (say) the name of the building in the picture, not even knowing if it is mentioned at all.

### Named descriptors

A more powerful scheme is the use of named descriptors: properties like title, location and time that have some more or less well-defined semantics associated with them. Using these, the user can describe the interval more precisely, and it is easier for the computer system to assist – the conceptual model is more complex, but the descriptions are more coherent and divided into smaller units. It is far more efficient and accurate to search for a particular location, whether manually or computer-supported, when it can be assumed can assume that it is either found as the value of the "location" descriptor, or not at all.

### Structured data values (objects)

To further increase the integrity and computability of the descriptions, it is possible to put constraints on the values of the descriptors. This may be as simple as demanding a specific date and time format for time descriptors, to prevent invalid values and make comparison and searching easier. Another possibility is to keep a list of possible values for a given descriptor – this avoids spelling problems, prevents invalid values, and may help create a faster and friendlier user interface for entering data.

Another improvement is to allow the data values themselves to be described. A location value may be a structured object, with its own descriptors specifying its name, its coordinates and its relation to other locations (contained in, adjacent to, north of etc.). This permits the creation of detailed, precise descriptions without redundancy – though the location may be referred to in many interval descriptions, its own properties are stored only once – but the model and the user interfaces supporting it necessarily becomes more complex.

### Type system / ontology

At the far end of the scale, we find models providing full-fledged type systems or *ontologies*. These provide formal methods of defining different types of data values, what properties they may have, and how they relate to each other. They may use mechanisms to define object types in terms of other object types, through the use of subclassing and composition, like in object-oriented programming languages. The same mechanisms may even be used for defining types of descriptors, and subtype/supertype relations between them.

This kind of model is very complicated to handle and understand for a user not educated in type systems and knowledge representation. Its main advantage is that it is possible to infer knowledge not explicitly present in the data corpus, by exploiting the semantics of the model. If the computer system knows the properties of the different kind of relations – whether they are transitive, reflexive, or symmetric, and what real-world phenomenon they represent, for instance – it can actually reason about the knowledge represented in the model

and come up with derived information. For this reason, such models are popular in artificial intelligence systems [Russel and Norvig 1995].

### 3.1.4 Flexibility

*Flexibility* is the ability to tune the content model to better conform to the user's wishes; to extend, change and adapt it according to the needs of the domain and the application. Again, this is a dimension of gradual improvements rather than discrete, easily identifiable steps, but a coarse classification may be defined as follows:

#### None

The conceptual model isn't flexible at all, beyond such possibilities as leaving parts of it unused, or using descriptors for other purposes than originally intended. However, this is detrimental to the quality of the data stored in the model. It can lead to inaccurate or misleading search and analysis of the data, if the user and/or the tools are unaware that expected elements of data are missing or stored in unexpected and counterintuitive slots.

Alternatively, a model may function well without any explicit flexibility, if it has been designed to be general. It may for instance contain only a few, very general descriptors that are applicable for nearly every purpose imaginable. However, the knowledge stored in such a model will necessarily be a bit bland and imprecise – the problems associated with weak structure.

Alternatively, a richer model may imitate flexibility by providing a host of optional descriptors, designed to cater for every need. This, though, leads to only a fraction of the possibilities actually being utilised, and the actual data drowns in the sea of unused structure, unless this flexibility is explicitly managed.

#### Managing descriptors

A simple way to introduce flexibility is to allow the user to manage – add and/or remove – descriptors. This does not necessarily affect the conceptual model as such, but it does affect the tools supporting it: They must provide a user interface for this customisation, and the tools for searching, browsing and analysis must be aware that the set of descriptors may change.

#### Managing object and descriptor types

A more complex and powerful flexibility stems from being able to manage object and descriptor types, not just the objects and descriptors themselves. Being able to create new object types, either from scratch or by extending or compositing existing types, enables the management almost any kind of knowledge about the video to be annotated. Creating new descriptor types by subclassing makes it possible to use very precise and specific descriptions, while still maintaining a degree of backward compatibility with tools that are designed to handle the less specific base descriptors.

### 3.1.5 Summary

In this section, I have presented four properties or dimensions for describing video content models: temporal expressiveness, spatial expressiveness, semantic expressiveness and flexibility. They may not be completely orthogonal – for instance, high flexibility, as I have defined it, is correlated to high semantic expressiveness. However, they are independent enough to not be redundant. I have not tried to establish exhaustive and uniform distance measures along them, but hopefully presented enough examples to make it meaningful to discuss models in terms of "high", "medium" or "low" flexibility or similar qualitative expressions. Some aspects of these properties may be as much or more tied to the tools and user interfaces supporting the model as to the model itself, but in my mind it is not fruitful to separate them too strictly. A model in itself is only an academic exercise; it needs a surrounding system if it is to be useful.

## 3.2 Existing video content models

This section presents a selection of video content models presented in literature, in chronological order of their publication. They are generally the most well-known and referenced examples of their kind, though a few more unknown designs have been included to show the width of the field.

### 3.2.1 OVID

OVID (Object-oriented Video Information Database [Oomoto and Tanaka 1993]) is a prototype system developed by Eitetsu Oomoto and Katsumi Tanaka. Part of their motivation was that meaningful scenes (temporal intervals) in a video document are identified incrementally and dynamically, according to various and changing user needs and domain requirements. It is not possible or desirable to *a priori* define an attribute schema that is suited for all kinds of scenes; each scene should be able to have an arbitrary attribute structure suitable for describing its contents. Also, descriptions should be reusable – if two scenes overlap, parts of their descriptions probably can, and should, be shared.

Figure 3.4 shows a UML rendition of OVID's conceptual model. The central concept is the Video-Object, which represents a "meaningful scene". What exactly constitutes a meaningful scene is left to the user to decide. The temporal extent of the Video-Object is denoted by a set of Interval objects – an OVID scene may consist of several disjoint video intervals. OVID doesn't consider multiple videos explicitly.

Video-Objects are described with arbitrary Attributes. Each attribute may take a value, and Values come in different flavours. The simplest is the Atomic Value, which is a string or a number. Video-Objects and Intervals may also act as Values, as may an arbitrary set of values.

Attributes and Atomic Values merit extra attention, as they provide the basic mechanisms that enable OVID's description reuse (or inheritance). Atomic Values may be defined as generalisations and specialisations of each other through an *is-a* relation, which is specified by users in advance. For instance,

"Japanese statesman" is-a "Statesman", and "Eisaku Sato" is-a "Japanese statesman". The relation is transitive, reflexive and anti-symmetric, and is used to infer the same semantics for other types of Values. For example, a Set Value is-another Set Value if each member of the first set is-a member of the other set. Based on this, OVID is able to reason about what is the correct set of Attributes and Attribute Values for unions and intersections of Video-Objects.

Attributes can be marked as *Inheritable*. This entails that if a Video-Object o1 contains another Video-Object o2 (meaning that o1 has no intervals outside of o2's intervals), o2 inherits o1's inheritable Attributes (with Values), unless the Attributes are already specified for o2. Together, these two mechanisms makes it easy to create new Video-Objects to increase or decrease level of detail, as they automatically are filled with correct descriptions, based on the Video-Objects they overlap. This can for instance be used for browsing.

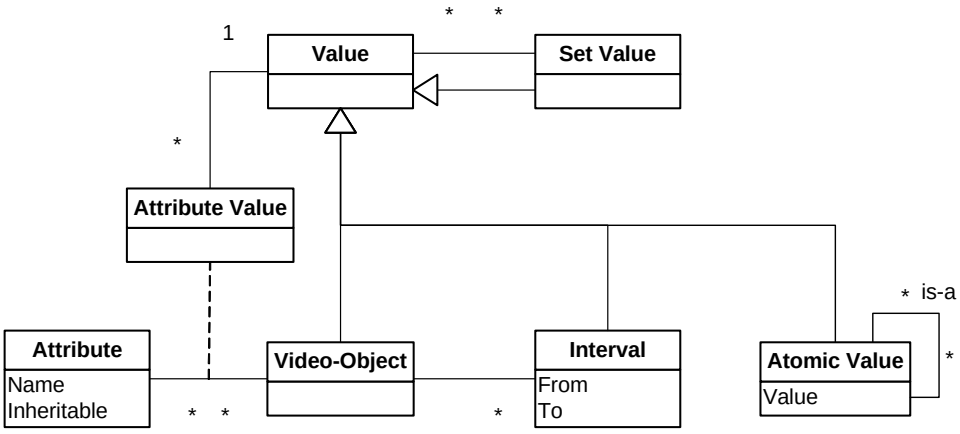


**Figure 3.4: OVID's conceptual model**

### Temporal expressiveness

OVID provides free overlapping intervals, as described in section 3.1.1. However, the paradigm of describing scenes (with aggregation and disaggregation) means that it is geared towards organising the annotations in a hierarchical segmentation. The prototype implementation provides a stratification-like graphical user interface for browsing (the *VideoChart*), with the Video-Objects as strata. It is questionable how useful this is, as the Video-Objects represent scenes, and thus will occur only once in each video (though they may contain multiple intervals, it is likely that these will not be very far apart). Recurring persons, locations and so on are stored as Attribute Values, and will not show up clearly in the VideoChart.

### Spatial expressiveness

OVID has no support for spatial annotations.

**Semantic expressiveness**

When it comes to semantic expressiveness, OVID's capabilities must be classified as "named descriptors", though "structured data values" could also be argued. The actual information in the system is in the end located in the Atomic Values, which cannot be described with anything but the is-a relationship. It is possible to indicate the presence of Eisaku Sato in different intervals, but the person Eisaku Sato cannot be described beyond the fact that he "is-a" "Japanese statesman". If a more detailed description (in the form of Attributes) is needed, he must be represented as a Video-Object, which may not make sense – he exists independently of any particular video intervals. However, the is-a relations form a taxonomy that is useful for querying, as well as for forming new Video-Objects based on others, as described above.

**Flexibility**

Likewise, "managing descriptors" is the most suitable class for OVID's flexibility. It is possible to create arbitrary Attributes, but not indicate what kinds of objects they should describe, nor what kind of values they can take.

### 3.2.2 Algebraic Video System

In [Weiss et al. 1995], Weiss et al. present a video conceptual model designed to cater for all aspects of digital video management: semantic and structural annotation; browsing and searching; editing, composing and reuse.

To accomplish this, they represent video as a tree structure with raw video segments as leaves, and algebraic operators as internal nodes. The algebraic operators include concatenation (one video segment is played followed by another), union (like concatenation, but common footage is not repeated), intersection (only common footage is played) and many others. The model permits arbitrary descriptions to be associated with the nodes; in the prototype, this is implemented as property-value pairs with textual values. Such descriptions are inherited by child nodes from their parent nodes.

Figure 3.5 shows the Algebraic Video Model, to the degree that it is representable in UML. There are almost 20 different operators; the most important – concatenation, union and intersection – have already been mentioned. The Create operator represents the leaf nodes in the tree model; it specifies a temporal interval from a particular video, and can have no children. Other notable operators include the Window operator, which displays the video represented by another node, in a particular region in on the screen; the Transition operator, which creates a timed transition like a wipe or a fade between two other nodes; and the Conditional operator, which plays one of several nodes, based on some condition. Thus, the model has the expressive power to define complex, interactive video presentations.

Figure 3.6 shows an example of the algebraic video model, compared to a stratified model. The empty nodes in the algebra tree represent Create operations, the $\cup$ node a union, and the $\circ$ nodes concatenation operators. One advantage of

the algebraic model is that it explicitly stores the relations between annotations – "Smith" is a child node of "Smith on economic reform" – which can be exploited in browsing and querying. In contrast, this information must be inferred from the temporal containment in the stratified model. However, it takes extra effort to build the tree (though it is unclear exactly how this is done), and it does not lend itself so readily to visualisation. In the prototype, browsing seems to be model-centric and not video-centric: The algebra tree can be traversed, and the corresponding video for each node played, but playing the video and thus finding relevant nodes and descriptions is harder to accomplish.

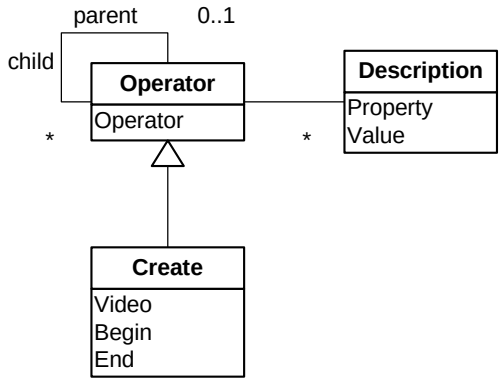


**Figure 3.5: Algebraic video model**

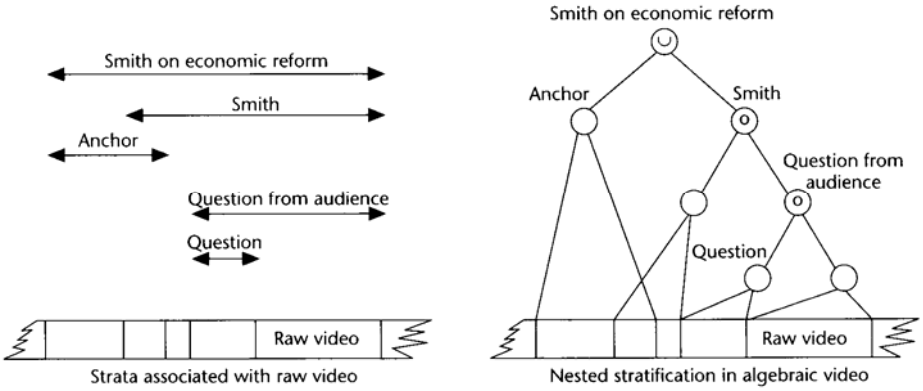


**Figure 3.6: Algebraic video example (from [Weiss et al. 1995])**

### Temporal expressiveness

With so many different operators for creating and describing video presentations, the algebraic model enjoys supreme temporal expressiveness. However, like OVID, the tree model will tend to favour annotations organised not unlike the hierarchical segmentation scheme.

### Spatial expressiveness

The algebraic video model has no support for spatial annotations.

**Semantic expressiveness**

The semantic expressiveness of the algebraic video model is limited to named descriptors. The video operators are very diverse and expressive, but they are concerned more with the structure of the video than its semantics. Moreover, unless the model is used for advanced video authoring, it is likely that only the most basic operators (Create, Concatenate and the set operators) will be used.

**Flexibility**

Likewise, the flexibility of the system is limited to managing descriptors.

### 3.2.3 VideoSTAR

VideoSTAR (Video Storage And Retrieval, [Hjelsvold and Midtstraum 1994], [Hjelsvold et al. 1995a], [Hjelsvold 1995]) is a database system developed at the Norwegian Institute of Technology (NTH, now subsumed into the Norwegian University of Science and Technology, NTNU). It proposes a comprehensive conceptual model designed to handle media files, virtual video documents, video structure and content-based annotations; and parts of it have been implemented.

Figure 3.7 shows VideoSTAR's conceptual model, adapted from [Hjelsvold and Midtstraum 1995]. The diagram has been somewhat simplified for clarity.

The upper left corner of the diagram concerns the video (and audio) files and the composition of video documents. The exact purposes of all the classes in the inheritance hierarchy are a bit unclear, but they enable VideoSTAR to handle both simple video files and complex, "virtual" video documents composited by fragments of several physical files. The CMObject (Continuous Media Object) is the abstract concept of a temporal media fragment, while a Stored Media Segment represents an actual media file.

VideoSTAR's temporal annotations are divided into two independent parts: structural annotations, and content annotations. The Structural Component and its subclasses enable a hierarchical segmentation of the video, with arbitrary depth. Each segment may have a textual description.

The content annotations are based around the Annotation class. This has several subclasses, corresponding to different kinds of concepts used to describe the video at hand. VideoSTAR provides four such classes – Person, Location, Keyword and Object – though it is noted that other classes should be constructed to fit the needs of particular domains and purposes. Annotations have descriptions; presumably, so do the concept classes.

Through the Stream Interval class, Annotations may be related to physical media files, or logical, virtual Video Streams. This enables reuse of annotations, since the same physical clip may be used in several video documents – annotations used to describe a Stored Media Segment are probably valid in most Video Streams containing the Segment.

VideoSTAR provides several tools for creating, browsing and searching the temporal annotations. It has a very fine-grained and expressive query language,

capable of formulating very complex queries on the temporal relationships of different intervals. However, the user interfaces never got past the prototype stage; they are complex, and not very good at visualising the temporal aspect of the annotations.

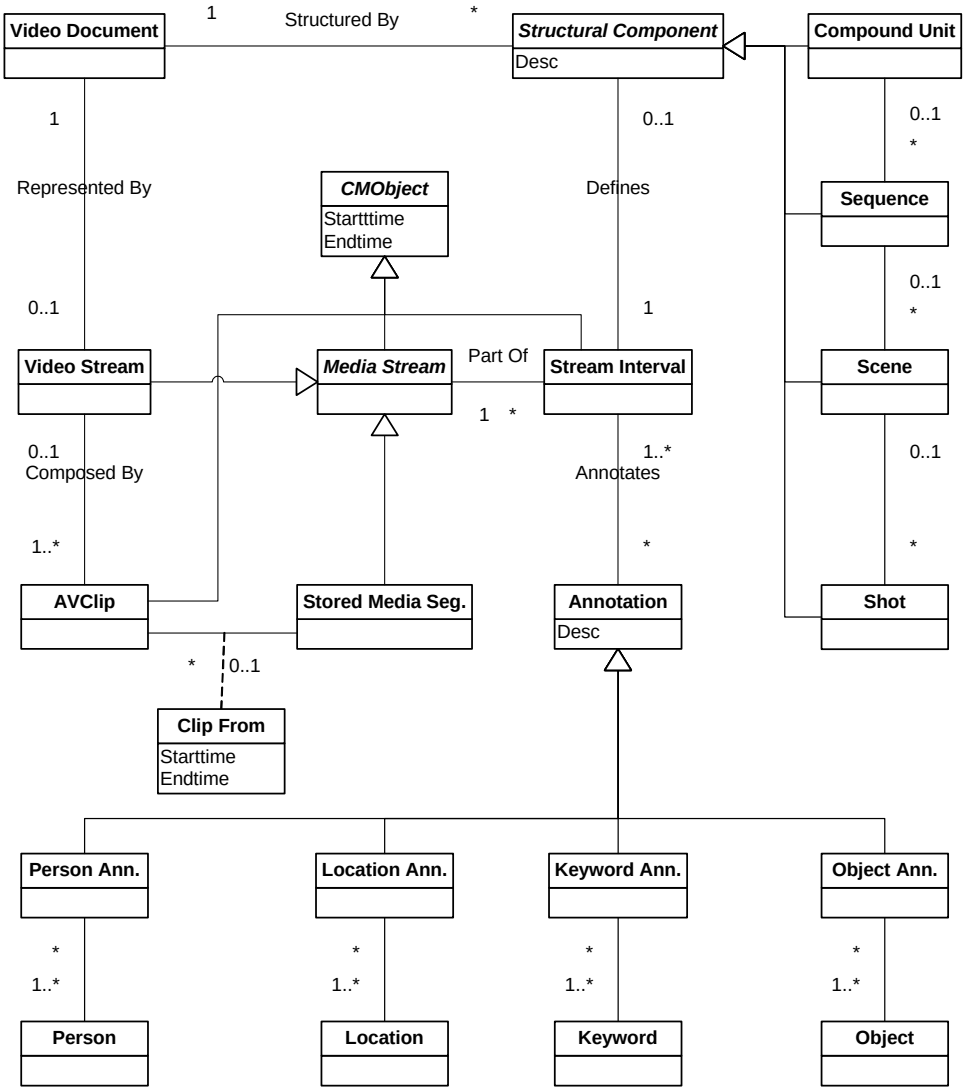


**Figure 3.7: VideoSTAR's conceptual model**

### Temporal expressiveness

VideoSTAR provides both hierarchical segmentation (with an arbitrary number of levels, no less) and free overlapping intervals with its Structural Components and Annotations, respectively. So in temporal expressiveness, it is second to none.

### Spatial expressiveness

VideoSTAR has no support for spatial annotations.

**Semantic expressiveness**

On the semantic expressiveness scale, VideoSTAR ends up on the weak side of structured data values (objects). Much of the information in the temporal annotations is kept in the free-text description fields, but VideoSTAR provides four classes of objects, enabling the user to describe (say) a person once, and connect him/her to all relevant intervals.

**Flexibility**

VideoSTAR isn't really flexible at all. Though the various VideoSTAR papers mention adapting the model by adding more annotation object classes, this must be done a priori, and affects the tools to the extent that they must be reprogrammed.

### 3.2.4 AVIS

AVIS (Advanced Video Information System) is a video database approach focusing on query processing. [Adali et al. 1996] presents a formal model for a video database, as well as index structures and algorithms for queries and updates.

Figure 3.8 shows AVIS's conceptual model. In AVIS's universe, two kinds of Entities occur in videos: Objects and Events. Objects are physical entities appearing in the video – people, for instance, or important items like the murder weapon in a crime movie. Events are more abstract entities, describing what is happening in the video – a murder, for instance. Both kinds of entities are described with a name field.

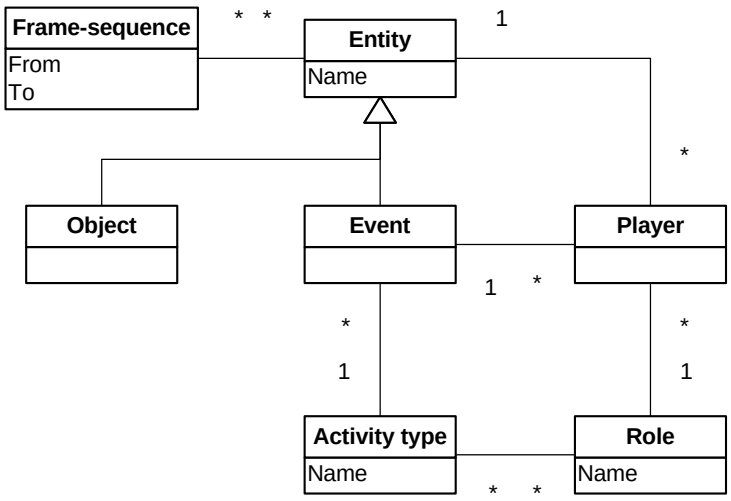


**Figure 3.8: AVIS's conceptual model**

Events are classified into Activity Types. For instance, "Murder" may be an Activity Type, and "The murder of John F. Kennedy" an Event classified as "Murder". In other words, Activity Types are abstractions of Events, and Events are instances of Activity Types.

An Activity Type has a set of Roles, described by name. A "Murder" Activity Type could for instance have the Roles "Murderer", "Victim" and "Murder weapon". Correspondingly, each Event has a list of Entities participating in the Event, and which Role each plays in the Event. Plain text strings can also play Roles in Events (e.g. for the Role "Murder motive", where neither an Object nor an Event may be appropriate), but this is omitted from the diagram for clarity.

To define the temporal properties of the Entities, each Entity is associated with a set of non-overlapping Frame-sequences. This creates a stratified annotation scheme, with the Entities as strata, as illustrated in Figure 3.9; AVIS calls this an association map. This seems to cause some redundancy. For instance, say that in the example above, the Event e1 has the Objects o1 and o2 as Players (in the Role "Murderer"). Thus, the presence of each Object in frames 2 to 3 is stored twice: once directly, and once implicitly through the Event annotation. It is unclear how this redundancy is handled in AVIS, but examples indicate that Events are ignored when searching for appearances of an Object. This could lead to incomplete query results, unless the system somehow forces consistency between the two ways to indicate the temporal properties of the Objects. Alternatively, there may be a semantic difference between being directly and indirectly related to a Frame-sequence; the fact that Events may have other Events as Players may indicate this.

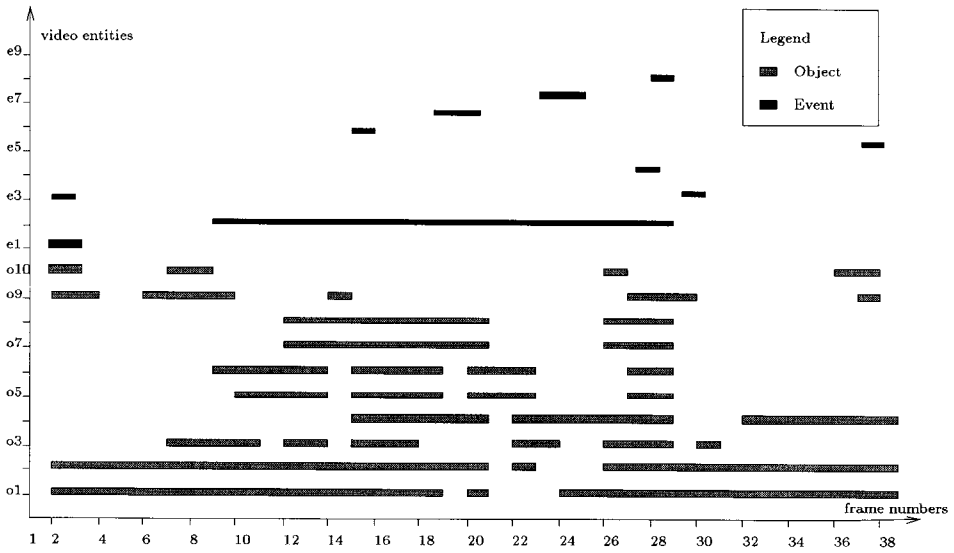


**Figure 3.9: An AVIS association map (from [Adali et al. 1996])**

One of AVIS's main contributions is the adaptation of a well-known spatial data structure, the segment tree [Samet 1990], to the temporal domain. The *frame segment tree* is a binary tree where the nodes are frame intervals, and the children are constitutes partitions of their parents. Figure 3.10 shows an example of such a tree, corresponding to the first ten frames of Figure 3.9. In addition to the frame intervals, each node also maintains a list of the Entities that are active in the frame

interval the node represents. Using this data structure, it is simple and efficient to determine which Entities are present in a given frame or frame interval. The abstract model presented in Figure 3.8 is for efficiency reasons implemented with the Entities referencing a linked list of segment tree nodes, instead of a set of frame intervals.

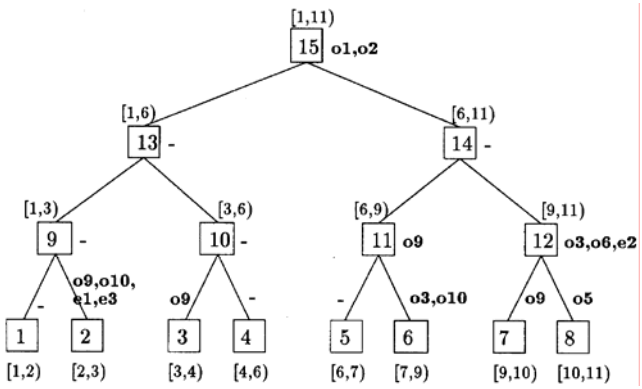


**Figure 3.10: An AVIS frame segment tree (adapted from [Adali et al. 1996])**

The model is designed to handle a single video, but the authors claim that it can easily be extended to handle multiple videos, by maintaining global indexes relating each Entity, Activity and Role to the video(s) they are used in.

### Temporal expressiveness

AVIS's association maps constitute free, overlapping intervals. As the number of leaf nodes in the frame segment tree (and thus the size of the tree, and the time spent traversing it during queries) is proportional to the number of distinct frame numbers where some Frame-sequence begins or ends, it is beneficial to restrict these endpoints to a limited number of "significant frames", thus creating a kind of segmented approach. But this is not a requirement.

### Spatial expressiveness

AVIS has no support for spatial annotations.

### Semantic expressiveness

AVIS is very minimalist when it comes to actual descriptions – the various elements of the conceptual model don't even have description fields, just names. Of course, AVIS is mainly an abstract model; if it were to be used in the real world, it is reasonable to assume that adding more descriptors would be a simple matter. Regardless, AVIS ends up in the "Structured data values (objects)" category, due to the composite nature of the Event Entity. AVIS provides a rudimentary classification functionality system through the use of Activity Types, but it is not powerful enough to be called a type system.

**Flexibility**

AVIS fits comfortably in the "Managing descriptors" category. A real-world implementation of AVIS would find it a simple matter to let users create new Roles and Activity Types, thus defining how their Events should be described. The flexibility is rather lop-sided, though; there is no flexibility at all in how Objects, Roles and Activity Types are described.

### 3.2.5 Vane

Vane [Carrer et al. 1997] is a video annotation engine developed at Boston University. It is a tool for semi-automatic production of metadata in the form of SGML [International Organization for Standardization 2001] documents, and is designed to be as open as possible for multiple domain-specific applications.

Figure 3.11 shows an overview of Vane's conceptual model; some attributes have been elided for clarity. Two kinds of temporal metadata are supported: structural metadata, and content metadata. The structural metadata includes segmentation of the video into shots, scenes and sequences – the common hierarchy mentioned in section 3.1.1. Shots are detected by image processing software; this is the automatic part of Vane's semi-automatic metadata production. Content metadata is modelled as generic objects with start- and stop-frames denoting the temporal interval in which they appear. The "Medium" attribute is used to indicate the mode of the object's presence: if it is shown in the picture, the Medium is "video" while it is "audio" if the object is only mentioned in the audio track. Objects may consist of a hierarchy of sub-objects, but the semantics of this is unclear. It is claimed that all the annotations belonging to a particular object can be grouped together to form a stratum, but it is also unclear how this is achieved, since the object and its description (including the temporal interval) is lumped together.

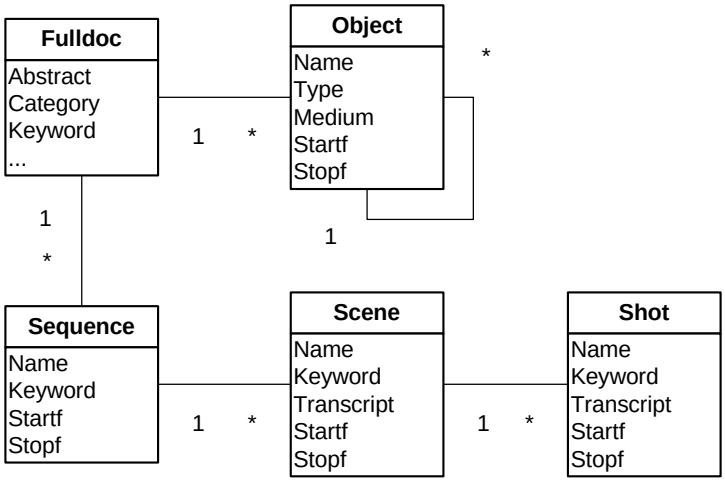


**Figure 3.11: Vane's conceptual model (simplified)**

In addition to the conceptual model described above, each element may refer to any other using generic SGML IDs and references, though apparently without any explicit semantics.

The model is SGML-based, and the model is specified using a Data Type Definition (DTD). This DTD may be extended to cater for different application domains. New attributes can be defined, and the order of attributes changed. Attributes such as Fulldoc's Category is designed to take a value from an *a priori* defined list; this list can be extended by modifying the DTD. These changes are reflected in the user interface without recompilation.

### Temporal expressiveness

Vane provides both free overlapping intervals (for content annotations) and hierarchical segmentation (for structural annotations). Figure 3.12 shows Vane's visualisation of its temporal annotations. The elements may be edited – segments can be joined, and endpoints moved by direct manipulation.

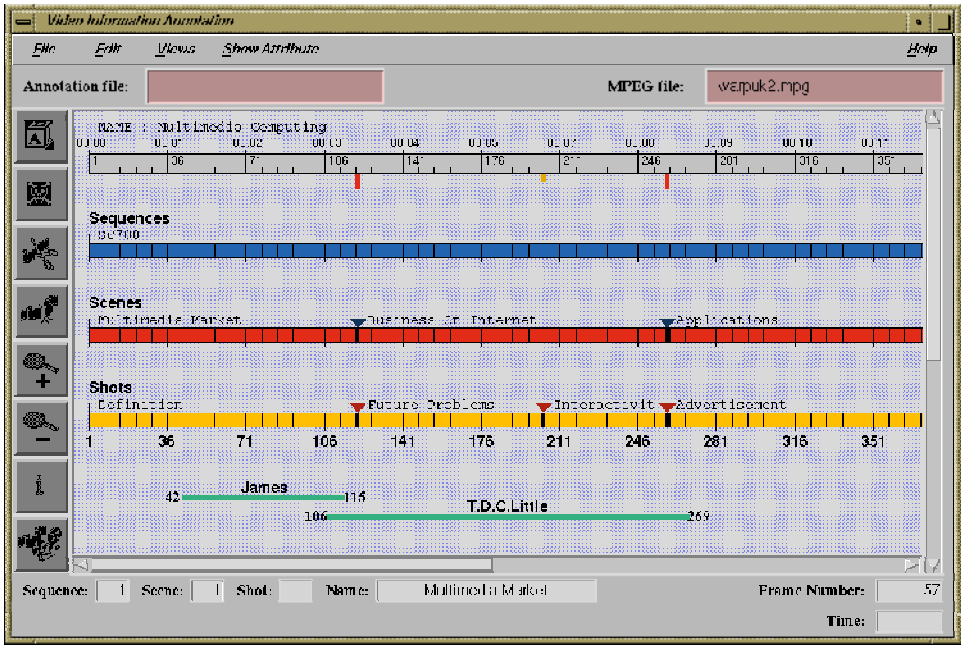


**Figure 3.12: Vane's temporal annotations**

### Spatial expressiveness

Vane has no support for spatial annotations.

### Semantic expressiveness

For semantic expressiveness, I put Vane somewhere between named descriptors and structured data values. Most descriptions are textual attributes, though a few (Fulldoc's Category, for instance) have their domains constrained to a list of values. Most elements can be annotated with references to other elements, but there are no semantics associated with these references, and no way to constrain what kinds of elements may be referenced. Figure 3.13 shows Vane's annotation window: The values are edited with either text fields or drop-down boxes.

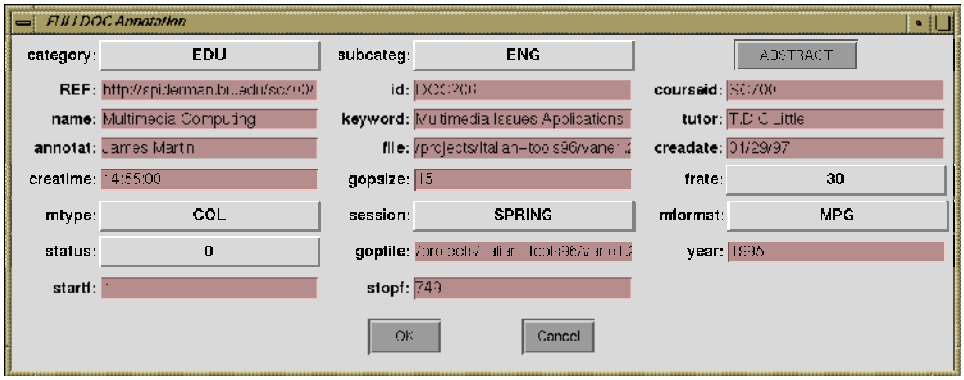


**Figure 3.13: Vane's annotation window**

### Flexibility

Vane provides facilities for managing descriptors. Attributes may be added to any element by extending the DTD. DTDs also provide mechanisms for defining new element types, but it seems that Vane does not take advantage of this.

## 3.2.6 Qualitative Media Analyzer

Qualitative Media Analyzer (QMA) [Skou 2003] is a commercial product, developed by Carl Verner Skou. As its name suggests, it is designed for qualitative analysis of interviews and observations, for research purposes. It is marketed as an alternative to making and annotating transcripts, with the advantages of saving time, keeping closer to the original material, and the possibility of computer-supported analysis of the annotations.

Figure 3.14 shows QMA's conceptual model. The Media Documents are media files; supported formats include Wave, Windows Media Audio, AIFF, MP3, MPEG-1, MPEG-2, Windows Media Video, AVI and QuickTime.

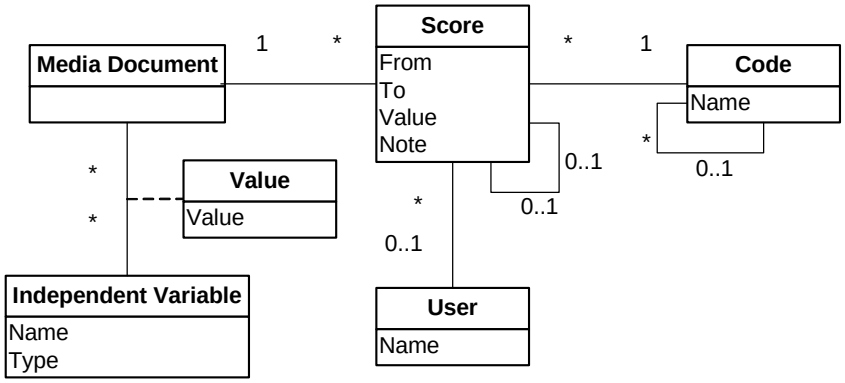


**Figure 3.14: QMA's conceptual model**

Media Documents are annotated by creating Scores. A Score is a temporal interval in a Media Document, related to a predefined Code. Scores have a start- and end-time with millisecond precision, as well as an integer value (the use of

which is unclear) and a textual note. A Score may also be linked to another Score, possibly in a different Media Document. Scores may, if desired, be associated with different Users, to facilitate the separation of different viewpoints on the same media.

Codes are in essence plain keywords. They may be organised in a hierarchy, but this is purely for visual organisation. The hierarchy has no semantic meaning – it is not used for aggregation during analysis, nor when selecting Codes for hiding uninteresting Scores, for instance.

A second annotation mechanism is the Independent Variables. These are named properties with a simple domain constraint – their value may be either text or an integer – associated with a Media Document as a whole. For instance, a recording of an interview may have the Independent Variables "interviewer" and "interviewee", with textual values (the names of the parties in the interview). Independent Variables, like Codes, are defined *a priori*, and are typically used for a collection of similar Media Documents.

QMA includes tools for organising Media Documents, for creating Codes and Independent Variables, for scoring (creating Scores), and for simple analysis of the Scores (number of Scores and total and average length per Code, for instance).

The analysis view provides some simple searching and browsing functionality as well, and given a set of Codes, Scores not related to them may be hidden in order to de-clutter the Score table. However, this functionality doesn't take advantage of the hierarchical organisation of the Codes, as mentioned earlier.

#### Temporal expressiveness

QMA provides free overlapping intervals, as described in section 3.1.1. It suffers from the user interface challenge this scheme entails, though – Scores are presented in a flat table, optionally sorted on start-time, as shown in Figure 3.15. This is not very user-friendly, especially if the number of Scores is large. There is a function for hiding the scores that are not active (that is, whose temporal interval doesn't contain the current playback time), but this mechanism is not dynamic; it only considers the playback time in the instant the function is selected.

#### Spatial expressiveness

QMA has no support for spatial annotations.

#### Semantic expressiveness

When it comes to semantic expressiveness, QMA falls in the "Named descriptors" category, as defined in section 3.1.3. The Codes act as a canonical vocabulary of terms for a project, and the user may define his or her own Independent Variables, with their simple domain constraints.

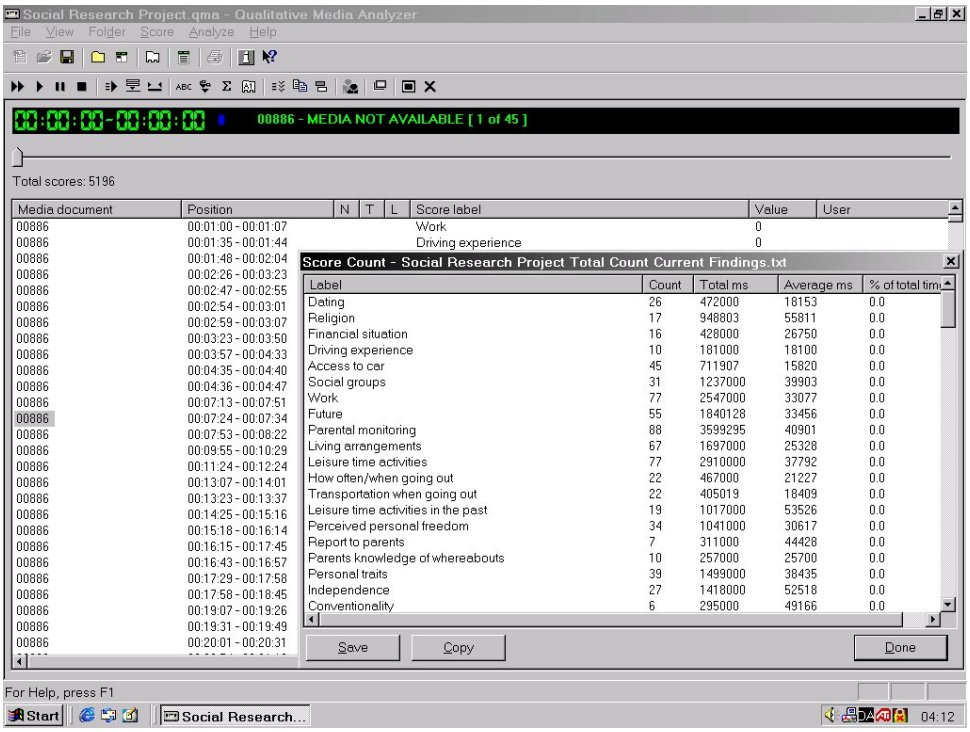


**Figure 3.15: QMA's user interface, analysis view**

### Flexibility

QMA provides facilities for managing descriptors, as discussed in section 3.1.4 – Codes and Independent Variables may be added, changed and removed by the user.

## 3.2.7 Noldus Observer

Observer [Noldus Information Technology 2003] is a commercial product developed by Noldus Information Technology. It is a competitor to Qualitative Media Analyzer, having the same purpose and target audience, and even a similar conceptual model and user interface.

Figure 3.16 shows Observer's conceptual model. It provides three methods for annotating video:

- Similarly to QMA, independent variables can be defined and used to describe a video as a whole (without any temporality). Variables can be numeric or textual, and constrained by minimum and maximum values (for numeric variables) or keyword lists (for textual).
- Time-stamped textual annotations. Note that these are connected to an instant, not a temporal interval.
- *Scores*, as described below.

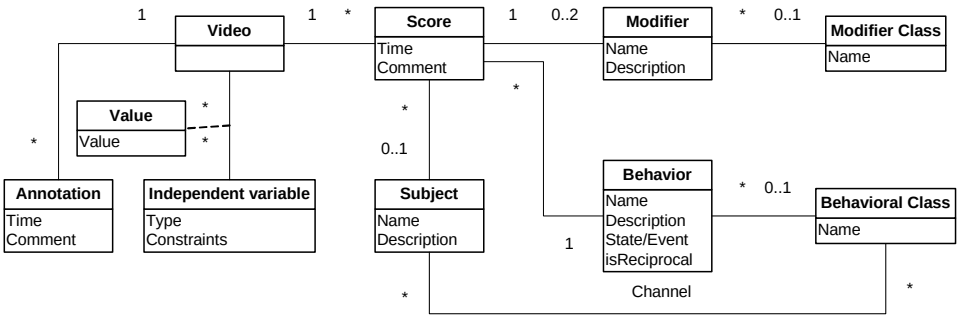


**Figure 3.16: Observer's conceptual model**

A Score has a timestamp, an optional textual comment, and a *Behavior*. Observer is geared towards behavioural analysis, so Behaviors are the main element in the model. A Behavior has a name and a textual description. Behaviors come in two flavours, States and Events:

- An *Event* is a Behavior where the duration is irrelevant or instantaneous; therefore, it has no more temporal properties than a single timestamp.
- A *State* is a Behavior where duration is relevant. It also has but a single timestamp, but it is considered to last until the next Score with a State in the same *Behavioral Class*. Behavioral Classes are used for grouping semantically related Behaviors. States within the same Behavioral Class are mutually exclusive (at least for the same Subject, though this is not completely clear from the documentation), and constitutes a partition of the video.

A Behavior may be labelled as *reciprocal*. A reciprocal behavior is symmetric; that is, it always works both ways. When an interaction in one direction is scored, e.g. Alpha plays with Beta, Observer will automatically create the reciprocal event, Beta plays with Alpha, if "plays with" is defined as a reciprocal Behavior.

A Score may also have a *Subject* (also known as "Actor"), denoting who is doing the Behavior – "Alpha" and "Beta" in the examples above. A subject and a Behavioral Class may be connected to form a *Channel*, a context where two States in the same Behavioral Class are not allowed to overlap. For instance, Alpha may be playing, eating or sleeping, but not at the same time; and Beta may be doing the same, but independently of Alpha. It is unclear how the mutual exclusiveness is handled when subjects are not involved.

Finally, a Score may have up to two *Modifiers*. These can be used to indicate the targets or objects of an incident ("Alpha throws rock at Beta"), the direction of something ("Beta runs away from Alpha") or similar. Like Behaviors, Modifiers can be grouped in Classes, but this does not entail any mutual exclusiveness.

Figure 3.17 shows a screenshot of Observer. Scores are shown in the "Event Log" with Actor (Subject), Behavior and comment; presumably the Modifiers are hidden because the user has chosen not to use them. The "Codes" window below shows the defined Behaviors (with their keyboard shortcuts).

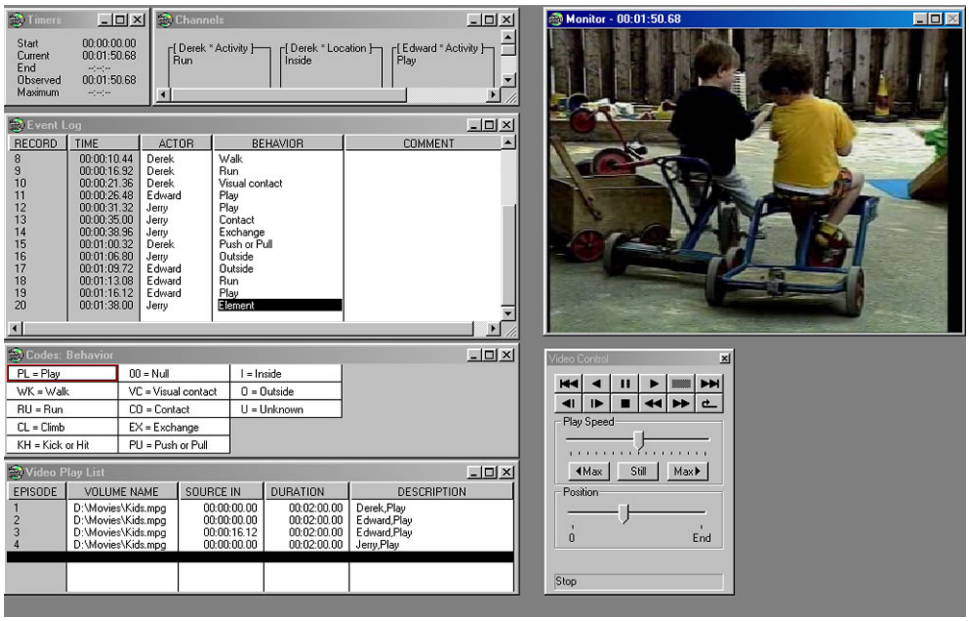


**Figure 3.17: Observer's user interface**

### Temporal expressiveness

Observer's temporal expressiveness is a bit odd. Its simplest temporal annotation functionality is free-text annotations, but these are connected to timestamps, not intervals. Thus, it makes no sense to talk about overlapping, and this scheme is perhaps closest to user-defined segmentation.

Events (or, to use Observer's terminology accurately, Scores connected to an Event) are also connected to timestamps, so the same goes for them. States, however, are interval-based, in that they are counted as valid from their timestamp to the timestamp of the next State in the same Class (or Channel). This constitutes a hybrid between user-defined segmentation and free overlapping intervals, since intervals in different Classes may overlap freely, but they form a partition of the video within each class.

### Spatial expressiveness

Observer has no support for spatial annotations.

### Semantic expressiveness

Observer fits best in the "structured data values" category. Its independent variables have type- and value-constraints, simple as they may be. Its temporal annotations, the Scores, have descriptors with reasonably well-defined semantics, and take values that are structured (albeit with nothing more than name, description and possibly a single-level classification). Behavior and Modifier Classes can be used for aggregation during analysis and presentation

### Flexibility

Observer lets the user define independent variables, Subjects, Behaviors (with Classes) and Modifiers (with Classes). This makes it easy to adapt Observer to a specific domain (playground interaction among children, or mating habits of baboons), but the basic semantics – behaviours of subjects – is unchangeable. It is for instance not possible to use the Score system to indicate the location or topic of a news video, without associating it with some Behavior. Thus, Observer ends up on the weak side of "managing descriptors".

## 3.2.8 Veggie

Veggie [Hunter and Newmarch 1999] is an application for describing video with Dublin Core-based metadata, developed at the state library and university of Queensland, Australia. Its purpose is mainly to enable quick, easy, cost-effective generation of standardised metadata that can be used to create online detailed visual summaries of videos.

Figure 3.18 shows Veggie's conceptual model. It is very simple: A video is segmented into scenes, and the video as a whole as well as each scene is described with appropriate properties. The properties are specified in an RDF schema (Resource Description Framework, [Brickley et al. 2004]), and Veggie's user interface for registering metadata is constructed according to this schema.

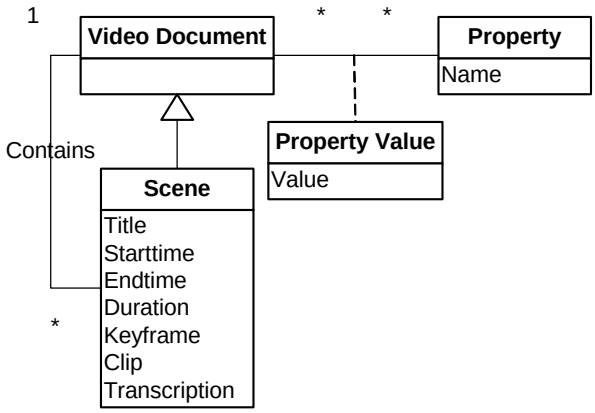


**Figure 3.18: Veggie's conceptual model**

Figure 3.19 shows a Veggie metadata form for the video level, using the Dublin Core properties. Figure 3.20 shows the corresponding form for the scenes.

Veggie stores the metadata in files, either in RDF [World Wide Web Consortium 2004a] or HTML format.

### Temporal expressiveness

Veggie's temporal expressiveness is limited to user-defined segmentation.

### Spatial expressiveness

Veggie has no support for spatial annotations.

### Semantic expressiveness

The semantic expressiveness of Veggie fits best in the "named descriptors" category. An RDF schema is used to define arbitrary properties; while it is possible to create data types, classes, objects and property constraints in RDF, Veggie doesn't seem to take significant advantage of this: The forms uses plain text fields for all properties.

### Flexibility

Similarly, the flexibility is best described as "managing descriptors". RDF provides support for objects and types, but it is apparently not used in Veggie.

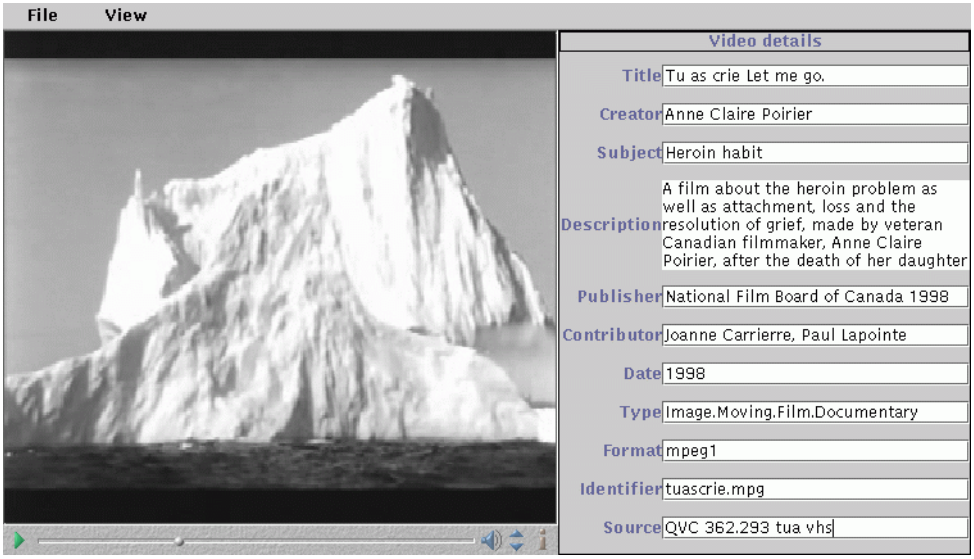


**Figure 3.19: Veggie's video-level metadata form**

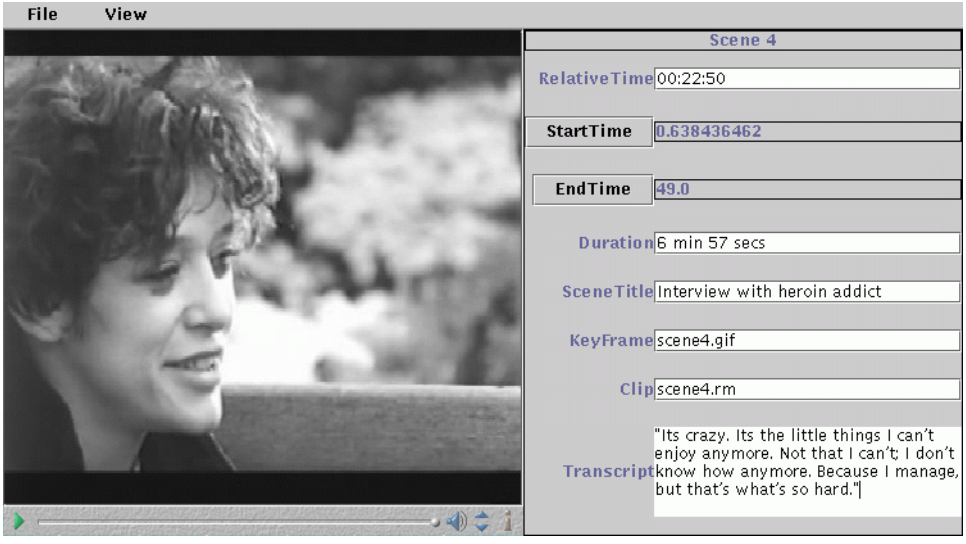


**Figure 3.20: Veggie's scene-level metadata form**

### 3.2.9 CARAT

CARAT [Hjelsvold et al. 1999] is a commercially available multimedia archiving system, developed by Siemens Corporate Research. It is designed to handle not only video, but also other media such as text and images, both as items to be described and as description values.

CARAT's conceptual model is not specified in detail, but a high-level overview is shown in Figure 3.21. It uses a two-level class system: Each object (or *item* or *media item*) belongs to an Object Type (e.g. baseball video), which again belongs to a Media Type (e.g. video). Objects have attributes, whose values are other objects. Attributes are defined by Log Structures, each of which is associated with an Object Type and a User Domain. In this way, objects of the same type are described homogeneously within the same domain, while allowing other domains to describe them differently.

An Object Type may be designated as a *controlled vocabulary* or *thesaurus*, with the effect that attribute values having that Object Type as their range must use one of the predefined Objects of that Type as their value. It is implied that such vocabularies may have an internal hierarchical structure as well (as the word "thesaurus" suggests; see appendix A), though this is not explained in detail.

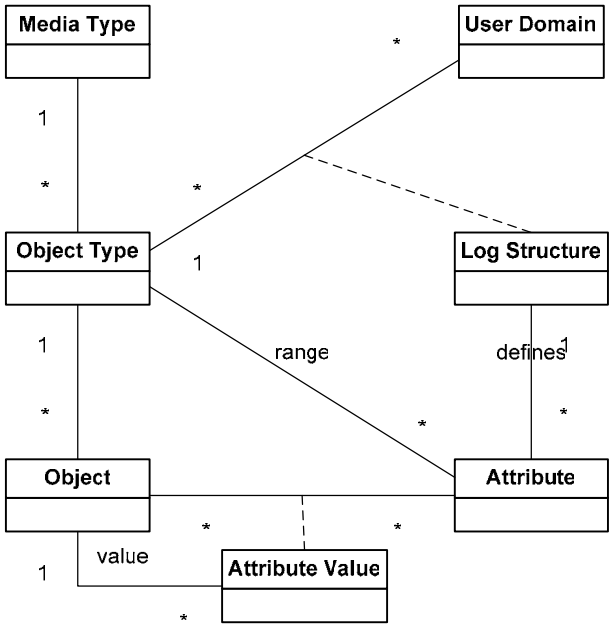


**Figure 3.21: An overview of CARAT's conceptual model**

The model also supports objects with a hierarchical structure (for video corresponding to hierarchical segmentation as described in section 3.1.1), though it is a little unclear how exactly this is done. In any case, the segmentation approach considers the shot as the smallest semantic unit of video. The system is quite clearly geared towards the commercial broadcasting domain.

**Temporal expressiveness**

CARAT provides hierarchical segmentation, apparently with no fixed number of levels, but with shots as the lowest level.

**Spatial expressiveness**

There exist plans to make it possible to define hierarchical image regions within images, but no further details are given. It is uncertain whether this also includes spatial regions within video.

**Semantic expressiveness**

CARAT provides a simple type system, allowing users (or at least DBAs) to define their own types and description schemes. This places CARAT at the high end of the "Structured data values (objects)" category.

**Flexibility**

Likewise, the type capabilities place CARAT in the low end of the "Managing object and descriptor types" category. Object types can be defined, though only in a quite simple manner. CARAT does not seem to support descriptor types.

### 3.2.10 BilVideo

BilVideo [Dönderler 2002], [Dönderler et al. 2003] is a video database system developed at Bilkent University in Ankara, Turkey. Its main contribution is perhaps the advanced, rule-based spatio-temporal modelling and querying functionality it provides, but it also includes more conventional temporal semantic annotations.

The spatiotemporal annotations is based on specifying minimum bounding rectangles (MBRs) for "salient objects" (objects the user is interested in) in each video frame. Based on these MBRs, the video is partitioned into segments. Within a segment, there is no significant change in the spatial relations between the MBRs, and each segment is represented by a keyframe.

Also based on the MBRs, spatial relations for each segment are extracted and stored as Prolog facts. For instance, if two objects A and B have MBRs as shown in Figure 3.22 for a segment in the video, several facts would be extracted and stored.

The fact *west*(A, B, *n*) would be stored (where *n* is the frame number of the keyframe representing the segment), since the centre of B's MBR is closest to the eastern line segment originating from the centre of A's MBR. BilVideo's Prolog knowledge base includes rules for inferring facts from other facts, so the inverse relation – *east*(B, A, *n*) – needs not be stored explicitly: there is a rule that says that if *west*(X, Y, *n*) holds, then *east*(Y, X, *n*) holds; similarly for the other directional relations. Thus, the number of stored facts is minimised.

BilVideo also supports topological facts like *cover*, *equal*, *inside*, *overlap* and so on. In the example above, *overlap*(A, B, *n*) is stored; *overlap* is defined as a

symmetric relation, so the reverse can be inferred without storing it explicitly. Likewise, 3D (z-axis) relations like *infrontof* and *samelevel* can be specified. This must be done manually, though, as the system cannot extract that information from the 2D MBRs. Finally, BilVideo stores the trajectory for each object, by creating lists of position (of the MBR' centre) and direction for each keyframe.

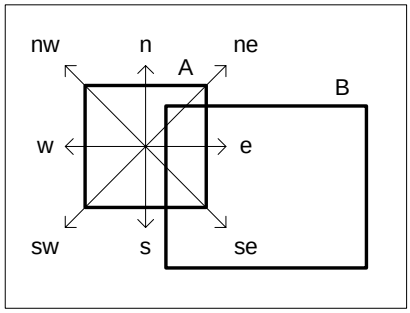


**Figure 3.22: BilVideo minimum bounding rectangles (adapted from [Dönderler 2002])**

Furthermore, supplementary high-level, domain-dependent rules may be added to the knowledge base, to describe objects or to infer events from the interplay of objects. For instance, for a football video, a goal could be defined as happening if the ball object is contained in a net object. Likewise, a pass could be defined as the ball being touched by a player object, followed by the ball being touched by a player object belonging to the same team, without any other player objects touching it in the mean time.

The semantic annotations functionality is designed quite differently. This is based on conventional relational database technology, with a conceptual model (slightly simplified) shown in Figure 3.23. Note the similarity to AVIS's model from section 3.2.4.

According to BilVideo, video consists of Events placed in time. Activities are abstractions of Events (or Events are instances of Activities); e.g. "wedding" is an Activity, but the wedding of Richard Gere and Julia Roberts in a movie is an Event. An Activity may have a number of Roles, e.g. bride, groom, best man and bridesmaid for the wedding Activity. Objects (for instance people) that appear during an Event may be assigned Roles for the Event through the Player class. Presumably, these objects are identical to those described in the spatio-temporal scheme discussed earlier.

Events may have subevents. For example, a "party" Event may have the Subevents drinking, dancing and talking. Objects may be assigned to Subevents as well as to Events, but Roles are only relevant for Events and Activities, not for Subevents and Subactivities. Subevents relate to Subactivities like Events relate to Activities; strangely though, there is no relationship between Subactivity and Activity.

Different Events may overlap temporally or leave gaps between them, as may the Subevents below each event. BilVideo calls the video intervals defined by Events and Subevents "sequences" and "scenes", respectively, though they do not imply

the strict hierarchical partitioning of the video normally associated with those terms.

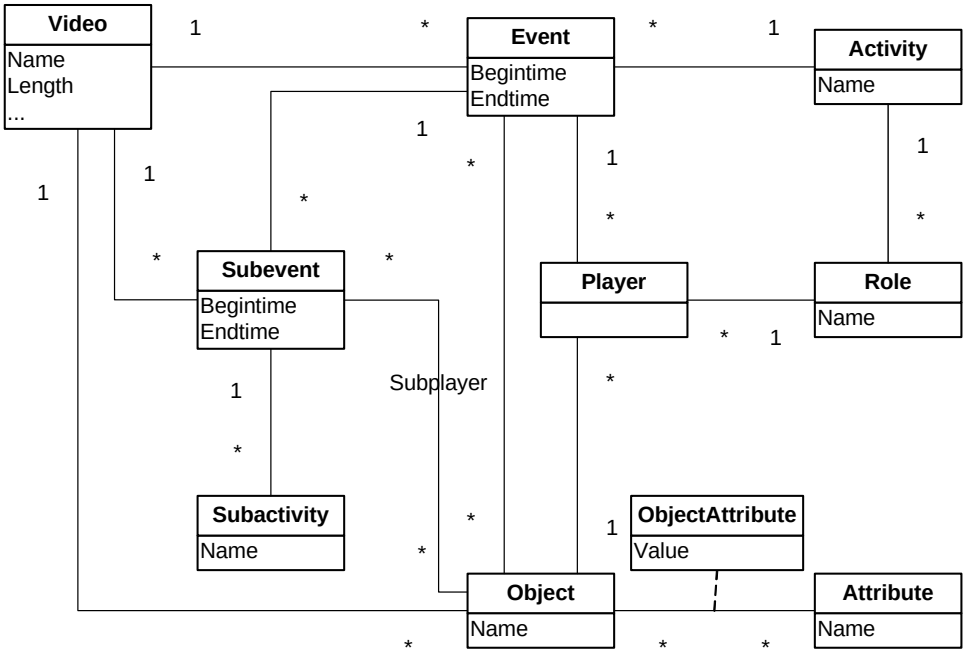


**Figure 3.23: BilVideo's semantic annotation model (adapted from [Arslan et al. 2002])**

### Temporal expressiveness

The spatio-temporal part of BilVideo uses user-defined segmentation. The video is partitioned based on the movement of the user-specified minimum bounding rectangles, so the user has at least implicit control over the segmentation. This leads to some redundancy – if the relationships between objects A and B are static, while their relationships to object C changes, the A-B-relationships must be duplicated across the segments caused by the movement of object C. [Dönderler 2002] notes that coupling relationships with frame intervals instead of keyframes would remove this problem, but make query processing significantly more complex.

The problem of partially invalid descriptions normally associated with segmentations does not occur in BilVideo, as the segment boundaries are determined by when the relationships between the MBRs occur. However, this leads to a large number of very small segments when there is much movement in the video.

It is unclear how well BilVideo handles *a posteriori* addition of new objects. This would in general necessitate resegmentation, as well as access to the MBRs for the existing objects, but it seems that MBR information (other than the MBR's centre's position at each keyframe) is discarded once segmentation and fact extraction has been performed.

The semantic annotations use free overlapping intervals. The hierarchies of Events and Subevents and Activities and Subactivities, as well as the classification of (Sub)Events as instances of (Sub)Activities seems eminently suited for a stratified visualisation, but BilVideo does not seem to take advantage of this opportunity. It should be noted, though, that BilVideo is geared toward querying, not browsing, as the access method of choice.

The semantic annotations alone have a rather poor expressiveness. A video of a wedding would contain a single Event – an instance of the "wedding" Activity – covering the entire video. During this time, one would expect to see a sizeable number of people appearing and disappearing from the screen, but in the semantic model, people (Objects) are related to an Event as a whole, not to their own little temporal intervals. The Subevents alleviate this, but that level may also be too coarse. However, presumably the objects in the semantic model are identical to those in the spatiotemporal, and so are related to fine-grained segments. Thus the semantic model becomes merely a way to specify more abstract, "non-tangible" annotations; aspects of the video that cannot be delineated with a bounding rectangle. A question remains, though: How is consistency between the spatiotemporal and the semantic model maintained?

**Spatial expressiveness**

BilVideo's spatial expressiveness ends up in the "dynamic regions" category. It is, however, quite limited in the kinds of areas it can represent; it uses bounding rectangles during registration, but stores only the centre points (in addition to the topological relationships). Also, BilVideo's segmentation scheme leads to some redundancy normally associated with consecutive static regions, as described in section 3.1.2.

**Semantic expressiveness**

The semantic expressiveness of BilVideo is a bit hard to classify. The semantic annotations fits best in the "named descriptors" category; though there is little other than name fields here, the Objects may at least be described with attributes and assigned roles.

However, as the spatio-temporal objects are defined in a Prolog environment, it is possible to define all kinds of predicates and rules to describe them, and so it is in theory possible to create a type system / ontology environment. This demands knowledge of logic programming and knowledge representation normally only found among specialised computer scientists, though.

**Flexibility**

The Semantic model includes the Attribute and ObjectAttribute classes, which gives the user (or the database administrator, at the very least) the power to define arbitrary attributes. This puts BilVideo in the "managing descriptors" class of flexibility.

Due to Prolog, though, BilVideo may be adapted with extreme flexibility. [Dönderler 2002] provides an example of rules and facts describing the football

domain, which enables a very detailed description of a football match, where passes, shots, ball control etc. is inferred by the spatial relationships using complex rules. But as mentioned above, such modification is beyond the reach of all but the most dedicated users.

### 3.2.11 Smart VideoText

The VideoText model [Jiang et al. 1997] is "a video data model based on the concepts of logical video segment and free text video annotations with arbitrary mapping between them". Its minimalist model is shown in Figure 3.24; it consists of unconstrained video intervals connected to free-text annotations through a many-to-many-relationship. A query language is also defined, supporting boolean operators as well as temporal ones (adjacent, during, overlaps etc.).

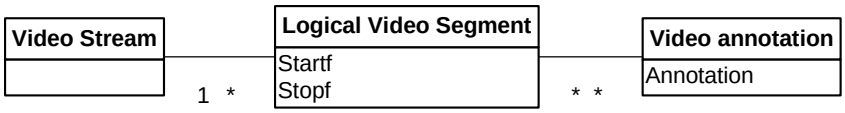


**Figure 3.24: VideoText's conceptual model**

Smart VideoText [Kokkoras et al. 2002] is an extension of this model; an attempt to utilise knowledge-based information retrieval techniques for video annotation. It uses essentially the same model, but introduces an additional element: a knowledge base, in the form of a conceptual graph [Sowa 1984a] [Sowa 1984b]. The purpose of this knowledge base is to enable reasoning about annotations and search queries – for instance, to have a query for "Greece" and "currency" match an annotation mentioning "European Union" and "euro", since Greece is a country in the EU, and the euro is a currency.

The knowledge base contains three kinds of knowledge:

- System knowledge, which includes rules about how to handle conceptual graphs – formation rules, inference rules etc.
- Domain knowledge, which includes knowledge not directly related to the video database – type hierarchies and such.
- Application knowledge, which is the knowledge directly related to the content of the video database. This is derived semi-automatically from the video annotations.

The video annotations are connected to concepts in the application knowledge with a many-to-many-relationship. Figure 3.25 shows this model.

This knowledge base is used during querying to improve precision and recall by substituting and disambiguating terms (concepts) according to the knowledge contained in the conceptual graph. A similarity measure between concepts is defined, based on the semantic distance (number of relation edges between concept nodes in the graph), and is used to rank search results.

The conceptual graph is also useful for browsing. This is achieved by using the semantic associations among the concepts video annotations and dynamically representing them as hyperlinks between the corresponding video segments. It is

debatable how user-friendly this is, however, since the conceptual graphs are not very easy to interpret.

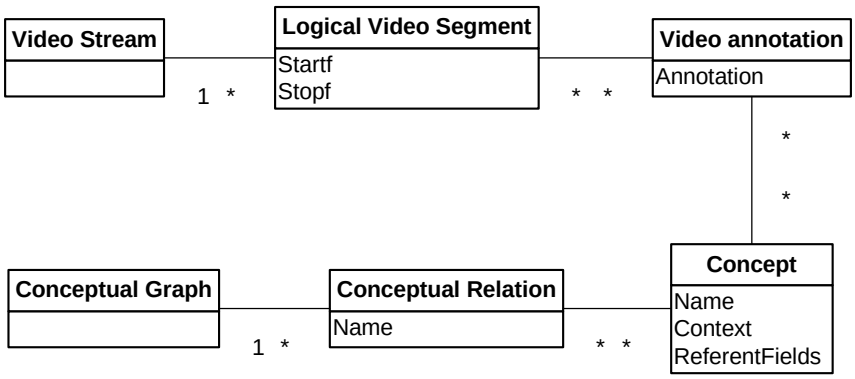


**Figure 3.25: Smart VideoText's conceptual model**

### Temporal expressiveness

Smart VideoText provides free overlapping intervals. It's possible that the Concept-relation may be used for stratification, but it seems that this model is not actually implemented, so it is hard to say.

### Spatial expressiveness

Smart VideoText has no support for spatial annotations.

### Semantic expressiveness

Smart VideoText provides a single, free-text description field (the Annotation field in the Video Annotation class in the diagrams above). Additionally, a conceptual graph is constructed semi-automatically from the Annotation, and its concepts related to the Video annotation. The conceptual graphs also contain pre-generated domain knowledge; supposedly this is constructed by hand in cooperation with domain experts.

The conceptual graph provides a type system/ontology framework, as described in section 3.1.3. However, it is unclear how this is used in practice, as Smart VideoText hasn't actually been implemented. Conceptual subgraphs are constructed from the free-text annotations, but how much control does the user have over this process? Is it possible to browse the graphs and connect concepts to the video intervals manually? What if a textual description is changed – are the concept connections revised automatically?

### Flexibility

The conceptual graphs give the user the power to define objects, object types, relations and relation types. This gives Smart VideoText the highest level of flexibility that I have defined. Of course, the user interface is important for determining if this power actually can be harnessed.

### 3.2.12 Discussion

This is hardly an exhaustive survey of all proposed video models and systems; they are far too numerous for that. However, the selection above represents the most common and influential designs in this field.

Many of these system focus a great deal on query systems, creating query languages with very detailed semantics. AVIS is a good example; it explains how it handles queries like "find all the people present in the scene when Brandon and Rupert discuss whether murder is a privilege reserved for a small group of people" (in the context of the Hitchcock movie "The Rope") [Adali et al. 1996]. To even consider posing a query like this, an intimate knowledge of the film in question would be needed. Also, to get satisfactory results from such exacting queries, the video would need to be described correspondingly accurately, and with the same point of view as the querier. Due to the inexactness of video semantics, and the unavoidable ambiguity in how it is represented in a structured model, this may not be likely. Furthermore, the level of detail with which different videos are annotated, may vary, leading to poor recall if the query is too specific. For these reasons, I believe such query systems and languages are a bit of a side track.

On the other hand, annotation-supported video browsing is in my opinion sadly neglected. There is less need for a complicated query language if individual video documents are easily browsed. Given a well-designed user interface, it would probably be quicker to locate the desired scene mentioned above by browsing, than it would be to formulate the query correctly; even more so if the user has some former knowledge of the video in question. Browsing systems are more robust to idiosyncratic annotation styles and variations in level of detail – the annotations are out in the open, not accessed indirectly through a wall of logic and syntax. Of course, some query functionality is required, but a simple keyword search in the vein of Google should suffice. OVID and the tree-based models provide some browsing support – traversing the tree structure, typically – but this is based on starting with a single node as a result of a very precise query, and wanting to view the context of its corresponding video segment.

On the third hand, video retrieval is but one possible application of video annotation systems. Another is video analysis, where the annotations act not merely as an index to the video, but as a knowledge base from which statistics may be generated, trends may be identified and new knowledge inferred. This calls for expressiveness, precision and formality – the model must be able to express knowledge about the video content concisely, unambiguously and in such a way that it is possible to reason about it. For such applications of video annotation, query languages may have their place[1] – at least in the theory about how the model can be manipulated; the actual user interfaces and tools is yet another matter. The point is that a proper video content model should be

[1] The already existing relational model and its algebra and calculus should suffice, though, with some suitable user interface built around it.

amenable to both kinds of interaction – browsing and analysis/querying – since they are both common, useful and complement each other. Searching for trends or patterns in video annotations might very well be done by visual browsing of some representation. Correspondingly, it might be useful to perform retrieval of relevant or interesting video sequences by formal and detailed specification of their properties, especially when these properties are "out in the open" – when the user can clearly see what kind of descriptions are used, and how they are used. A good video annotation system (and its model) should support both paradigms, and not separate them needlessly.

Figure 3.26 shows a graphical representation of the characteristics of the described video content models, according to the dimensions of temporal expressiveness, flexibility, semantic expressiveness and spatial expressiveness. The horizontal lines signify levels of "full" expressiveness or flexibility; a system that had full score in all the categories would have a column four lines high. The most obvious feature of this diagram is that full temporal expressiveness is very common, and that spatial expressiveness is very rare. Smart VideoText and BilVideo are the most powerful models (it could also be argued that the semantic expressiveness of Smart VideoText is at least as high as that of BilVideo), but they are probably also the ones that are hardest to use and understand, due to their dependence on logic programming. CARAT is more conventional in that regard, and scores highly, but has a somewhat limited temporal expressiveness.

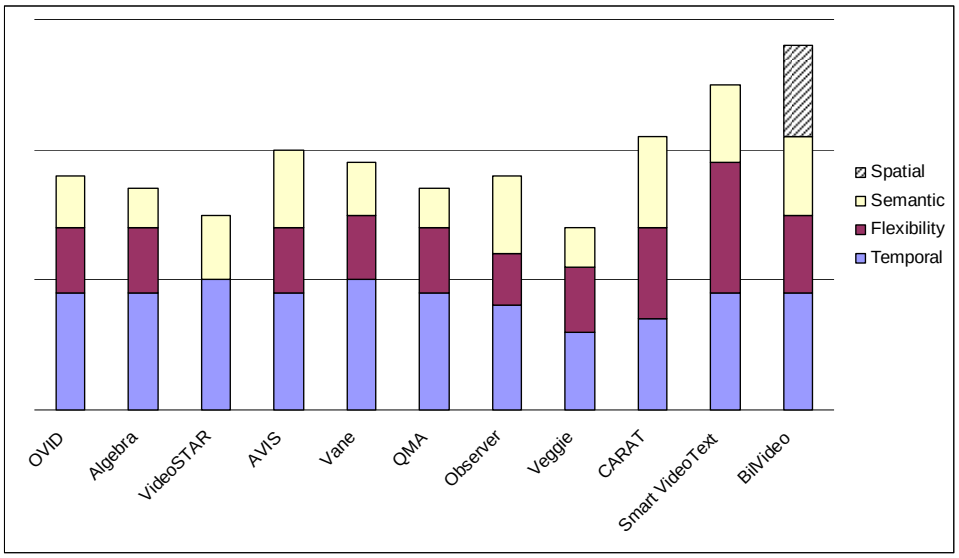


**Figure 3.26: Video content model features**

It is interesting to note that almost none of the models have more than half score in flexibility and semantic expressiveness. As noted in [Oomoto and Tanaka 1993], video is so diverse that it is almost futile trying to create a single, fixed model and description scheme to cater for all applications. However, the common practice of using generic "objects" and "events" for everything makes for bland semantics. A system focusing on these two aspects might help bring the state of

the art a little further. Usability should be taken into account, though: Requiring changes in the database schemas to adapt the model for specific purposes (or perhaps worse, requiring a working knowledge of logic programming) is not user-friendly enough. A different approach is needed.

## 3.3 Existing tools and user interfaces

One might perhaps expect well thought-out annotations models to be accompanied by similarly well-constructed user interfaces and tools, and vice versa, but that is seldom the case. Many of the systems and models presented in the previous chapter provide little or no user interface at all. Those that do – the commercial applications, for instance – rarely have the most interesting properties. The reason, of course, is that conceptual modelling and user interface design are two separate disciplines, so scientific papers (and the work behind them) focus on one or the other.

Hence, this chapter presents an overview of tools and user interfaces for video annotation – systems for producing, visualising, searching and/or browsing annotations – with correspondingly little focus on the underlying models. This topic is more diverse than that of the previous chapter, so it is not practical to try to classify these efforts according to a fixed set of dimensions or properties.

This is not an exhaustive survey of video database applications, since quite a few of them aren't particularly interesting within the context of this work:

- Some have very simple conceptual models, typically a simple segmentation scheme, where the segments are treated like documents in a conventional digital library. Since the focus in this thesis is on more expressive semantics, these are mostly ignored.
- Some focus not on metadata, but on presenting and searching in low-level features of the video (images, colour distribution, or, in the more high-level cases, closed-caption text), or on creating abstracts of video using keyframes, pause removal and similar techniques. Again, the focus of this thesis is on high-level, semantic metadata.

Instead, I have tried to find examples of the most typical, interesting and/or original tools and interfaces for production and retrieval of video content annotations.

### 3.3.1 Marquee

Marquee [Weber and Poon 1994] is a tool for real-time video logging. Through a pen-based interface, users can segment the video, annotate the segments with handwritten text and illustrations, and apply user-defined keywords to the segments.

Figure 3.27 shows Marquee's main interface. In the main note taking area to the right, new segments (called "time zones") are created by drawing a horizontal line with the pen, linking the segment to the current playback time. Handwritten notes

are linked to the time zones they are written in. Thus, a mapping between the temporal and the spatial domain is established.

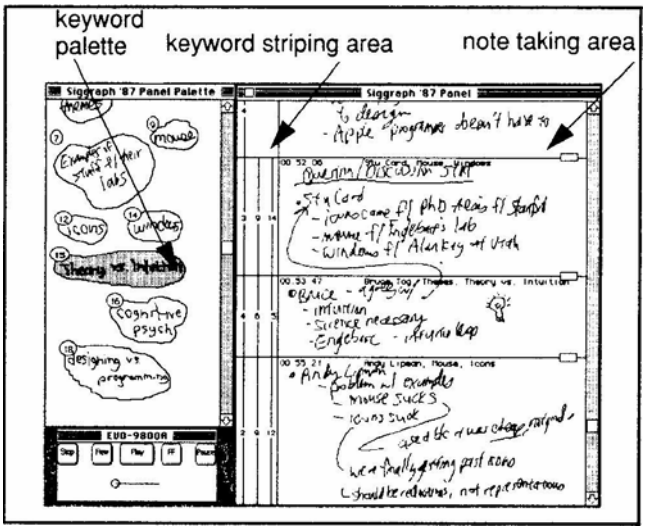


**Figure 3.27: Marquee's user interface (from [Weber and Poon 1994])**

Circling a word or phrase defines it as a keyword, assigns it a unique ID number, and copies it to the keyword palette to the left. Then, by clicking on a keyword and drawing a vertical line in the keyword striping area, one or more keywords may be assigned to one or more time zones. The area in the lower left corner is used to control video playback.

One of the main foci of the study was to determine if useful logging could be done in real time, which was confirmed; though not to the degree that it was practical to participate actively in a meeting and log it at the same time. Additionally, several questions concerning issues such as retrieval and the impact of individual note-taking styles were left unanswered. As the notes themselves are very unstructured (idiosyncratic and in some cases even illegible), retrieval has to be based on the keywords, which can fairly easily be made machine-readable; however, the time zones present a useful, non-temporal overview suitable for intra-video browsing.

### 3.3.2 Audio Notebook

The Audio Notebook [Stifelman et al. 2001] is a somewhat similar system. It is a cross between a digital audio recorder and a paper notepad, shown in Figure 3.28. Like Marquee, free-form annotations are made in real-time using a pen, but here they are written on real paper. However, the Notebook timestamps each pen stroke, linking it to the corresponding instant in the audio being recorded. Sensors in the Notebook keep it aware of which page is being written on, and it even handles multiple notepads.

After recording/logging, the user may access any part of the recording by pressing the pen on a note; this will start playback from the time the annotation was entered. As an aid to navigation, a LED strip to the left of the notepad acts as a

scrollbar, showing the playback position within the current page, and can also be used for random access within the page.

Qualitative user studies show that the Audio Notebook not unexpectedly has several advantages over both regular notebooks and plain audio recorders for browsing and review. Its retrieval capabilities are rather limited, though, as the annotations are not machine-readable.

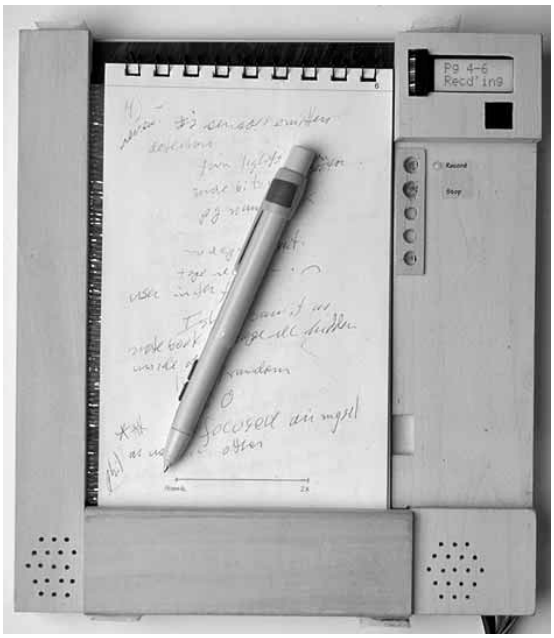


**Figure 3.28: The Audio Notebook (from [Stifelman et al. 2001])**

### 3.3.3 Logjam

Logjam [Cohen et al. 1999] is another tool for real-time annotations, but with a bit more structured annotations. It produces free overlapping intervals, each assigned to a category (stratum) and described with a text string, as shown in Figure 3.29. In this interface, the user can create new intervals and categories, edit their descriptions, and control video playback. However, the main contribution of LogJam is its tangible logging interface, shown in Figure 3.30.

This consists of a board capable of detecting hardwood blocks containing small digital chips. Users indicate the presence of locations, events, people and behaviours by placing and removing the blocks representing these categories on the board, which is connected to the logging computer. Several people may log concurrently, using the same board.

The system proved successful in making video logging a group process, though technical problems with the tangible interface detracted from the accuracy of the system. However, the logging speed was not increased. Though the loggers worked in parallel, whatever time this saved was spent in discussions, or waiting on the slowest logger. Also, for single-person logging, the keyboard-and-mouse

interface was preferred, because the board was cumbersome and inaccurate, and the user disliked to switch between it and the keyboard, which was needed anyway for descriptions, category definitions and editing. But the logging process itself was deemed more social and fun when using the board in a group.

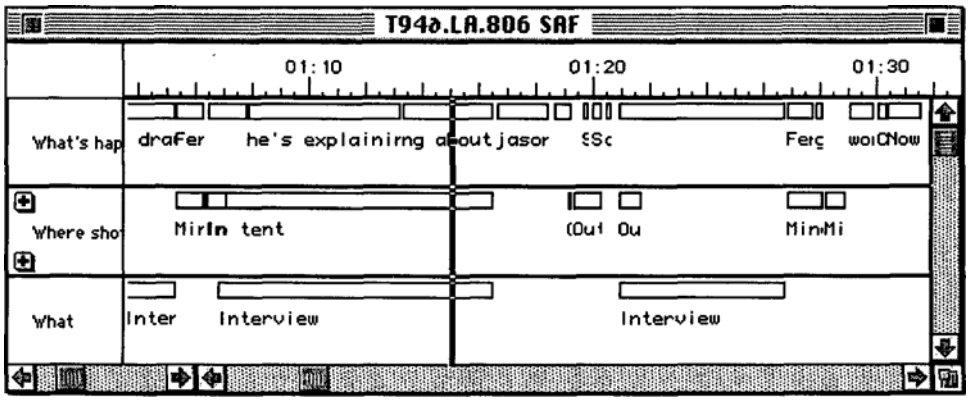


**Figure 3.29: LogJam on-screen user interface (from [Cohen et al. 1999])**

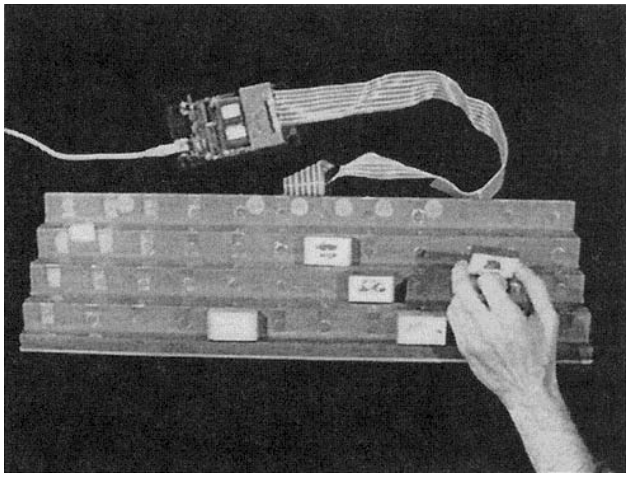

**Figure 3.30: LogJam logging board (from [Cohen et al. 1999])**

### 3.3.4 Video-based retrieval

[Gordon 2000] describes a system for semi-automatically retrieving information relevant to video segments, primarily for educational purposes. A video is manually annotated with codes indicating relevant topics for different intervals in the video. Based on the interval borders, the video is segmented into homogenous clips. For each clips, the topic codes are used to gather information from one or more online digital libraries, using the Z39.50 protocol. The resulting information is used to construct web pages that are displayed alongside the video, synchronized with the video according to the segment boundaries, as shown in Figure 3.31. The letters refer to various functions of the interface; A: the video; B: segment description; C: categories of related material (overview); D: descriptions of related material; E: hyperlinks to the actual related material.

The system is quite useful for enhancing the educational experience of watching video, though the semi-automatic approach produces a relatively high amount of irrelevant information. However, the learning experience is inherently very linear, since playing the video is the only way to access the extra information. Additionally, the information retrieval is one-way, from video to annotations, and the system offers no search or browse capabilities beyond jumping around in the video with the playback controls.

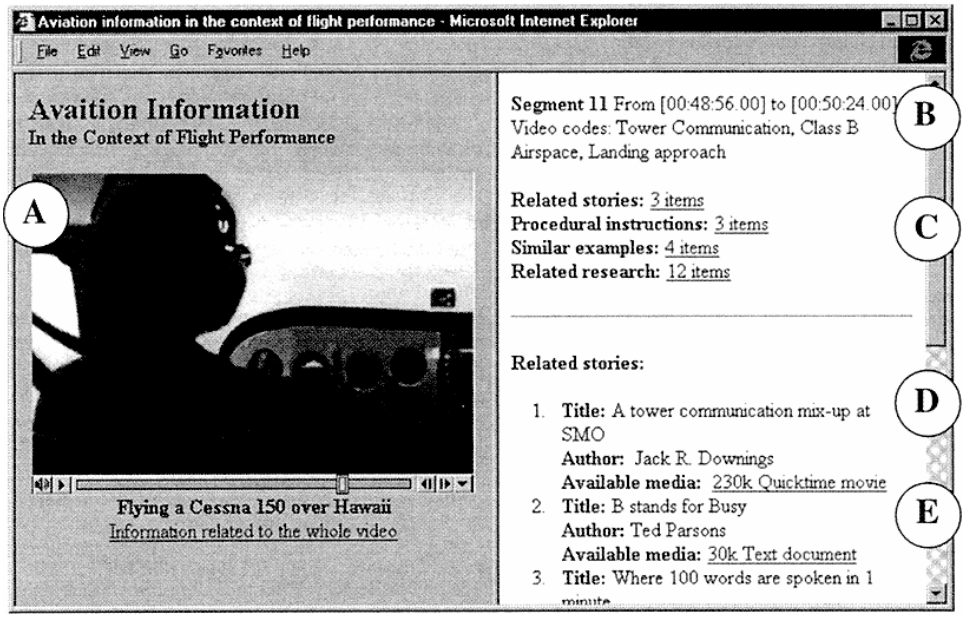


**Figure 3.31: Gordon's video annotation interface (from [Gordon 2000])**

### 3.3.5 Rframes

Rframes [Arman et al. 1994a] is one of many browsing systems based on representative frames (or keyframes, as they are often called). In this case, the video is segmented into shots, and from each shot a single frame is selected. This frame is then augmented with visual elements suggesting the length of the shot, and motion indicators based on the movement of the border pixels for each frame of the shot. This is illustrated in Figure 3.32.

By displaying these Rframes in chronological order, a visual index is constructed, allowing the user to quickly get an overview of the content, as well as the facilities to jump directly to shots of interest. The motion indicators are in addition helpful in identifying scene changes overlooked by the segmentation algorithm. An Rframe can also be used to search for other, similar Rframes (and thus similar shots, hopefully), based on colour distribution and moment invariants – but the Rframes cannot really be called semantic annotations, since they deal only with the visual properties of the video. Another weak point is that the Rframes are rather big in terms of screen space, so they are not particularly suitable for browsing more than one video at a time.

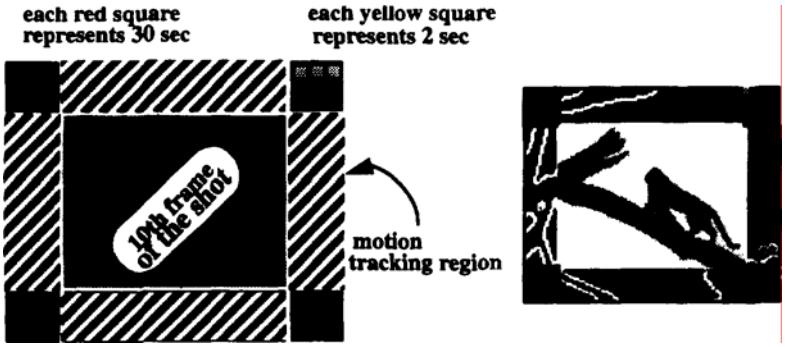


**Figure 3.32: Rframe composition; example Rframe to the right (adapted from [Arman et al. 1994a])**

### 3.3.6 MSR Video Skimmer

The MSR Video Skimmer [Li et al. 2000] is an extension of Microsoft's Media Player, with several interesting features:

- It can remove pauses (i.e. segments without sound) from the video.
- It can play the video at different speeds, using signal processing to preserve the pitch of the audio.
- It segments the video into shots, presents shot boundary frames, and provides controls for jumping to shot boundaries.
- It supports a simple textual table of contents for the video, as well as personal notes entered by the user.

The Video Skimmer is shown in Figure 3.33. This is an interesting tool, though the annotation model is very simple (after all, it is not the main point of this work). It is only suitable for intra-video browsing, but the techniques it utilises are interesting candidates for inclusion in a more complete video management system.

### 3.3.7 Hierarchical Video Magnifier

The Hierarchical Video Magnifier [Mills et al. 1992] is another keyframe-based video browser. In this approach, a video is represented as a timeline with tiny thumbnail pictures at regular intervals, on which is placed a resizable, movable magnifier. The section of the video covered by the magnifier is shown below the timeline as a sequence of frames taken from regular intervals, as shown in Figure 3.34.

In this way, the user can easily zoom in on interesting sections of the video, without losing awareness of the surrounding context, and jump around in the video just like with a traditional timeline slider. Clicking on the second-level timeline creates another magnifier, so the user can zoom in even further, while preserving the first-level magnification. This creates a hierarchy of timelines and magnifiers, as illustrated in Figure 3.35, where three additional levels of magnification have been created.

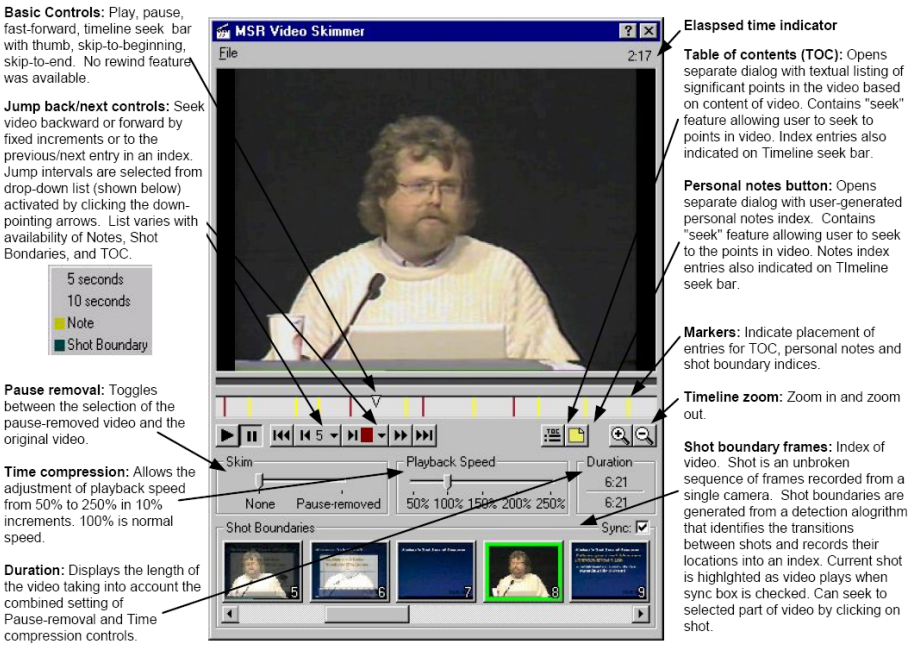


**Figure 3.33: MSR Video Skimmer (from [Li et al. 2000])**

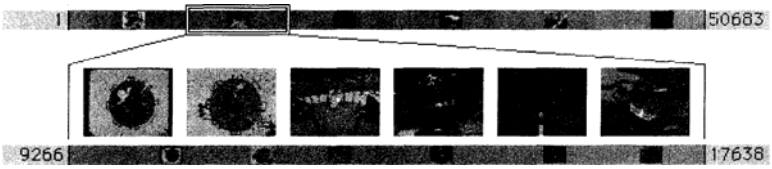


**Figure 3.34: Hierarchical Video Magnifier, one level (adapted from [Mills et al. 1992])**

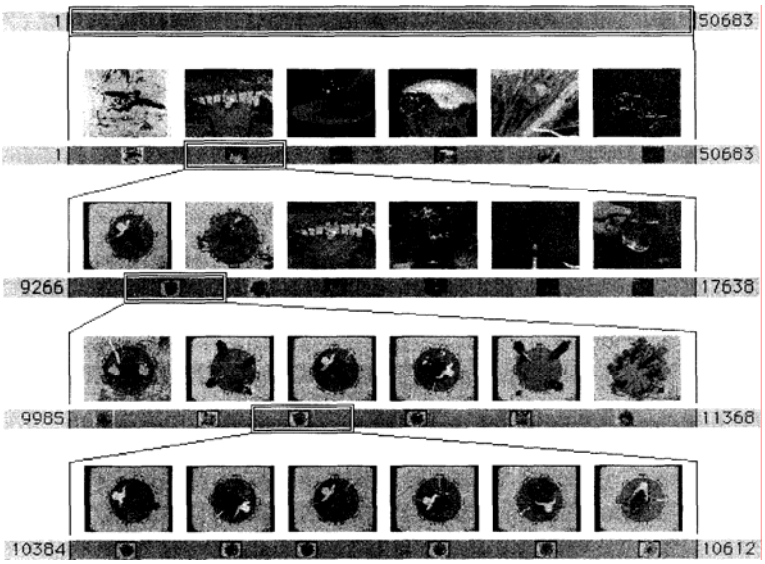


**Figure 3.35: Hierarchical Video Magnifier, four levels (from [Mills et al. 1992])**

The magnifier excels at providing details (keyframes, at least) while preserving overview, but it has no search capabilities or annotation functionality. Also, like most video browsing systems, it is not practical for browsing more than one video at a time.

### 3.3.8 Jabber

Jabber [Kazman et al. 1996], [Kominek and Kazman 1997] is a system for indexing and retrieval of video conferences. It performs an analysis of the audio recording, extracting nouns to use for index terms. Using a thesaurus, it groups the nouns into clusters of related meaning, thus creating a smaller set of topics or concept, each represented by the lowest common hyponym. It also performs an analysis of the pattern of interchange between the speakers, to determine the type of the discourse (or the "temporal idiom", as the authors call it): Presentation, Question and Answer, Discussion, Argument or Agreement.

Figure 3.36 shows Jabber's user interface. Each participant in the conference has a time line, where the recognized nouns are displayed. At the bottom of the window, a table displays the concepts and the nouns that comprise them. Selecting a concept causes the corresponding words to be highlighted in the time lines. Selecting another highlights its words in a different colour.

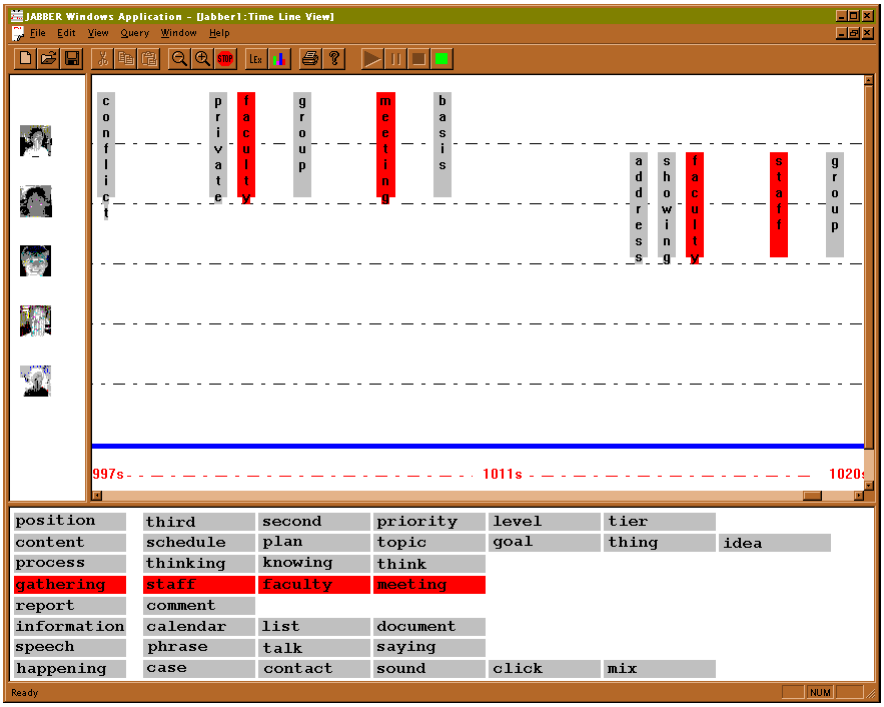


**Figure 3.36: Jabber**

Jabber seems like a very useful and practical tool, especially since it can perform its indexing unassisted (except for setting it up, deciding thresholds and parameters and such). Its method for conceptual clustering makes it quite robust, since it can successfully disambiguate words by noting in which context they

appear. It doesn't depend on individual words, so perfect speech recognition is not crucial. However, the usefulness of the generated index concepts may be questioned – it is not obvious what the conference in Figure 3.36 was all about ("priority handling of work requests", as it happens), or what exactly was discussed at various points. The user interface (and the time line display in particular) may be another weak point – the figure shows just twenty seconds; an overview of the entire conference would most likely render the word indicators even more illegible than they already are.

### 3.3.9 VoiceGraph

VoiceGraph [Oard 1997], [Slaughter et al. 1998] is a retrieval interface for speech data, shown in Figure 3.37. It is based on automatic transcription of the speech, and search is initially performed using standard text information retrieval techniques, with ranked results. However, the speech is also analysed to determine "alternation patterns" (who speaks when, like the "temporal idioms" of Jabber above), whether the speaker is male or female, and (if possible) speaker identification. Automatic language identification and music detection is also mentioned, though it is unclear if this is actually implemented. In any case, this information is represented graphically for each speech document matching the search criteria. In the figure below, male voices are rendered as blue line segments, while female speakers are red. The idea is that this display can be used to determine the relevance of the search result, as the user can quickly see the pattern of the discourse in the speech document; how many speakers there are, how they take turns and so on. Users can also select individual segments, and get the transcription displayed in the bottom of the window, and play the selected segment.

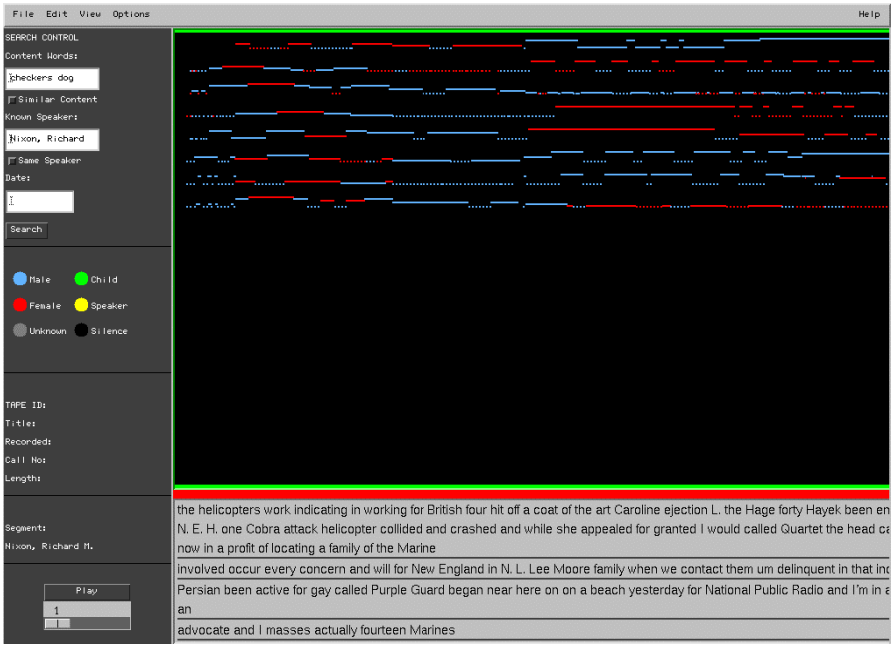


**Figure 3.37: VoiceGraph**

User studies showed that the alternation patterns were not an easy concept for novice users to grasp, but that they had a definite potential as an aid to browsing. The system is tailored to a rather specific domain, but it is a significant strength that no manual annotation is needed.

### 3.3.10 SCAN

SCAN, Spoken Content-based Audio Navigation [Whittaker et al. 1999] is also a speech retrieval interface, with a goal similar to VoiceGraph's. The systems segments speech documents into "paratones" (speech paragraphs), and performs automatic speech recognition to create a transcript for each paratone. Textual information retrieval techniques are then used to create a ranked list of speech documents, based on a user's query.

The main contribution of SCAN is its visualisation and navigation interface, shown in the centre of Figure 3.38. This shows a timeline view of the selected speech document, with paratones indicated by the short, vertical black markers, and each paratone's relevance to the query is indicated by a vertical column in a kind of histogram. The height of the column indicates the relevance of the paratone, and it is also colour-coded in order to show which of the query terms that match. Selecting a column highlights the corresponding transcript paragraph in the text area below, and clicking on a paragraph plays the corresponding paratone.

Like its name suggests, SCAN's primary purpose is to help users scan audio documents, and quickly determine their relevance to a query. It succeeds at this; a user study shows (not surprisingly) that it made fact finding and relevance judgment significantly easier, compared to a simple audio player without any visualisation or transcripts. However, it did not help matters much when users were asked to create summaries of the speech documents. The authors offer several possible explanations for this result: that search terms often were evenly distributed throughout each document, so no paratone distinguished itself as a candidate for summarisation; that highly relevant paratones used synonyms for the search terms, thus not showing up in the histogram; and that the quality of the transcription was too poor.

SCAN is obviously a useful tool, though it has some limitations. The visualisation only shows the search terms, so it is not very suitable as an overview of the content of a speech document as a whole. For this, the transcript must be used, but this quickly becomes very large, and has no structure apart from the division into paragraphs/paratones. It shows only one speech document at a time, making it difficult to compare them with each other. It also has trouble with synonyms, but this could of course be remedied with a thesaurus and more intelligent information retrieval techniques. The quality of the automatic speech recognition is also a major factor in the usefulness of SCAN, but it is of course a great advantage that no manual annotation is needed.

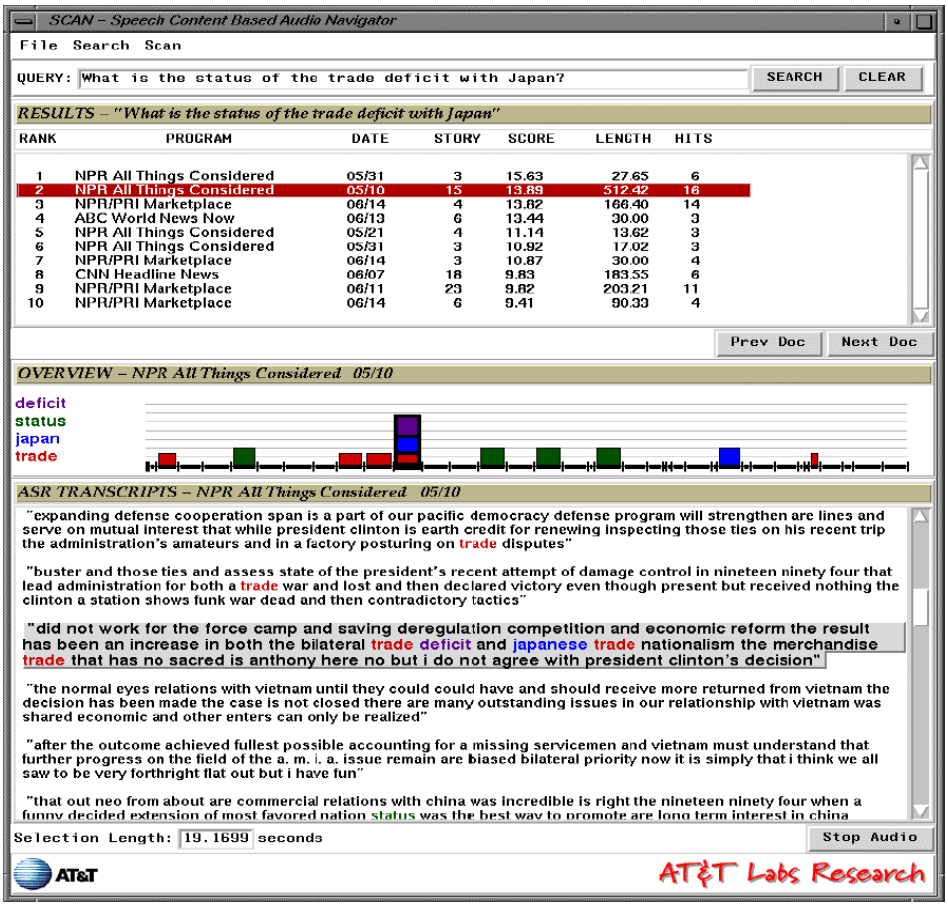


**Figure 3.38: SCAN**

### 3.3.11 Media Streams

Media Streams [Davis 1993] is, according to the author, "an iconic visual language for video annotation". The idea is to manually create semantic annotations using icons, in order to end up with a language- and culture-independent description of the content of a video. The system includes over 6000 icons, arranged in a hierarchical structure – the icons dealing with "space" include among others icons for land, North America, USA, south-western USA and California. This makes it easier to browse for appropriate icons during annotation, and is presumably useful for retrieval as well.

Annotation is performed by dropping icons on a timeline, as shown in Figure 3.39. The timeline is organised into 44 streams (strata), each dealing with a certain aspect of the video: settings, characters, objects, actions, camera motions and so on. Each iconic description is considered valid from its insertion point to the next scene break, thus providing a temporal expressiveness somewhere between free overlapping intervals and structure-based segmentation. Icons may be combined to construct compound icons, expressing relatively complex semantics like "Janet is positioned to the right of a shrub".

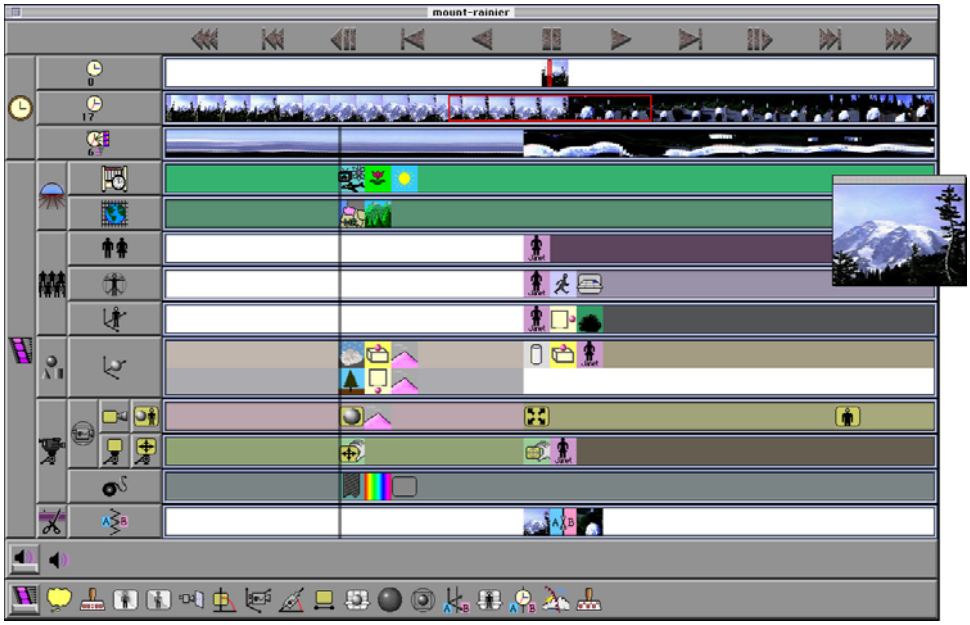


**Figure 3.39: Media Streams timeline**

Media Streams is an interesting idea, especially its language independence. However, the semantics of the icons is not always obvious to an untrained user. They also need quite a lot of space in order to be legible, so there is reason to think that an overview of an entire hour-long video will be very cluttered. The fixed number and semantics of the 44 strata/streams seem a little restrictive, though one would assume that they probably cover most aspects of video one might want to annotate. Finally, it seems extraordinarily labour-intensive to annotate video in this way.

### 3.3.12 DIVA

DIVA [Mackay and Beaudouin-Lafon 1998] is a system for "exploratory data analysis of multimedia streams", with the purpose of enabling users to visualise, explore and evaluate patterns in data that change over time. Several streams are defined, each dealing with one aspect of the video to be analysed. For instance, a video documenting the behaviour and work processes of air traffic controllers might be annotated with streams indicating when a controller is speaking to another, when one is using the radar, when two controllers are writing on flight strips simultaneously, and so on. This is in essence a stratified annotation scheme with free, overlapping intervals.

Figure 3.40 shows DIVA's "streamer display". This plays the video in the middle, while the streams are displayed as coloured ribbons moving from the lower right corner to the upper left corner as the video plays. Thus, both the "past" and the "future" states of the streams are displayed with time mapped to the spatial domain. The semantics of each stream is indicated by the letters above and to the left of the central video display. In the illustration, The "T", "D", "S" and "Notes" streams are active at this particular moment, while the "PW" stream is just

coming up. The “Notes” streams are text streams – they can have textual descriptions associated with each interval they contain.

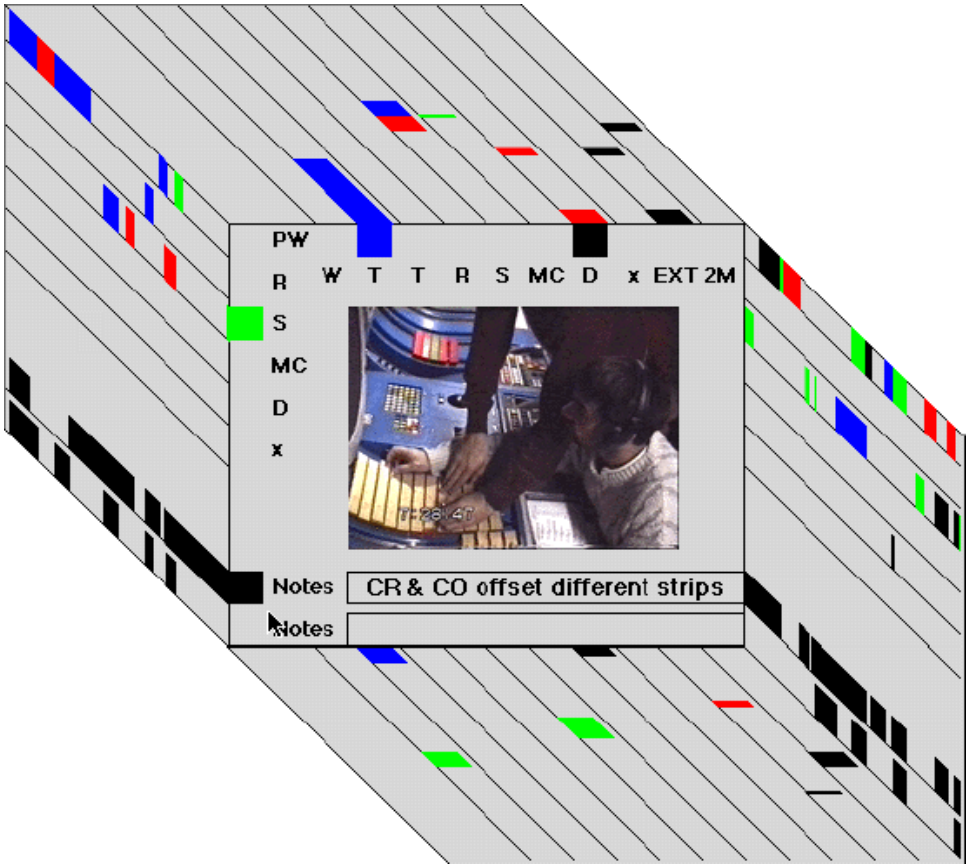


**Figure 3.40: DIVA**

DIVA also provides a stream algebra to perform editing, change the visualisation and search the streams; and a direct manipulation user interface for editing streams.

Informal evaluations show that this representation gives a better perspective on the data, compared to a spreadsheet display. However, for general-purpose browsing and retrieval, it seems to have a few drawbacks. It is very space-intensive, and shows only one video at a time (or two synchronised, but not with independent annotation streams). Also, the use of only one or two letters to identify streams may be a little too terse. To display streams both on the “floor”/”roof” and the “walls” of the streams display makes it somewhat difficult to see the temporal relation between them.

### 3.3.13 MMVIS / TVQL

MMVIS (MultiMedia Visual Information Seeking) and TVQL (Temporal Visual Query Language) [Hibino and Rundensteiner 1995], [Hibino and Rundensteiner 1996], [Hibino and Rundensteiner 1997], [Hibino and Rundensteiner 1998] are

an analysis tool and a query interface for temporal data. The TVQL interface consists of four sliders specifying the relationships between the start- and end-points of two temporal intervals, shown in Figure 3.41. The particular configuration shown, specifies that interval A and B must start at the same time; that A must end when B ends, or up to 4.8 seconds before; that A must end after B starts, but no more than five seconds after; and that A must start between five seconds before and 4.8 seconds after B ends. These four relationships aren't independent – if for instance A and B start at the same time, B cannot possibly end before A starts. TVQL handles this by ensuring that while one slider is manipulated, the other three are changed accordingly (though strangely, this seems not to be the case in the figure below).

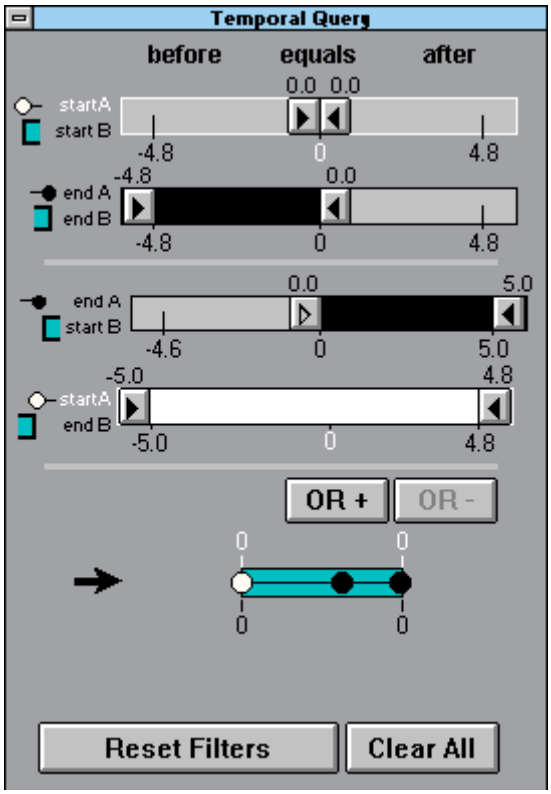


**Figure 3.41: TVQL**

In any case, this interface is capable of representing each of the thirteen temporal relations specified by Allen [Allen 1983], and it can also specify subsets of them, due to the use of ranges instead of just saying that one interval boundary must be "before", "equals" and "after" another. User studies have shown that while TVQL is somewhat difficult to learn, it is significantly faster and more accurate in use than a forms-based interface.

MMVIS is an analysis tool "where users can browse video data in search of temporal trends by specifying temporal queries via direct manipulation". The main paradigm is to select two subsets of the annotation intervals, query for temporal relationships between the subsets, and view a visualisation of the result,

in a dynamic and iterative manner. In MMVIS, each annotation interval is described with four properties: name, action, receiver and category. Subsets of intervals are created by specifying sets of values for these properties. Then, a temporal query is made using TVQL, and the result is visualised as in Figure 3.42. This shows an analysis of a beach volleyball game, where the annotations intervals are described with name (the names of the four players – Kiraly, Dodd, Steffes and Whit), action (serve, dig, pass, set, hit, kill and block) and category (team red, team black, point play to red, side-out rally to black and so on). In this example, the two annotation subsets are A: those where name is "Dodd" or "Whit", and action is "serve"; and B: those where name is "Kiraly" or "Steffes" and action is "dig" or "pass". The subsets are visualised using yellow circles for A, and blue squares for B, and the area of the symbol corresponds to frequency of the intervals. Thus, it is easily seen that Kiraly passes more often than Steffes, but digs and serves are equally distributed among the players.

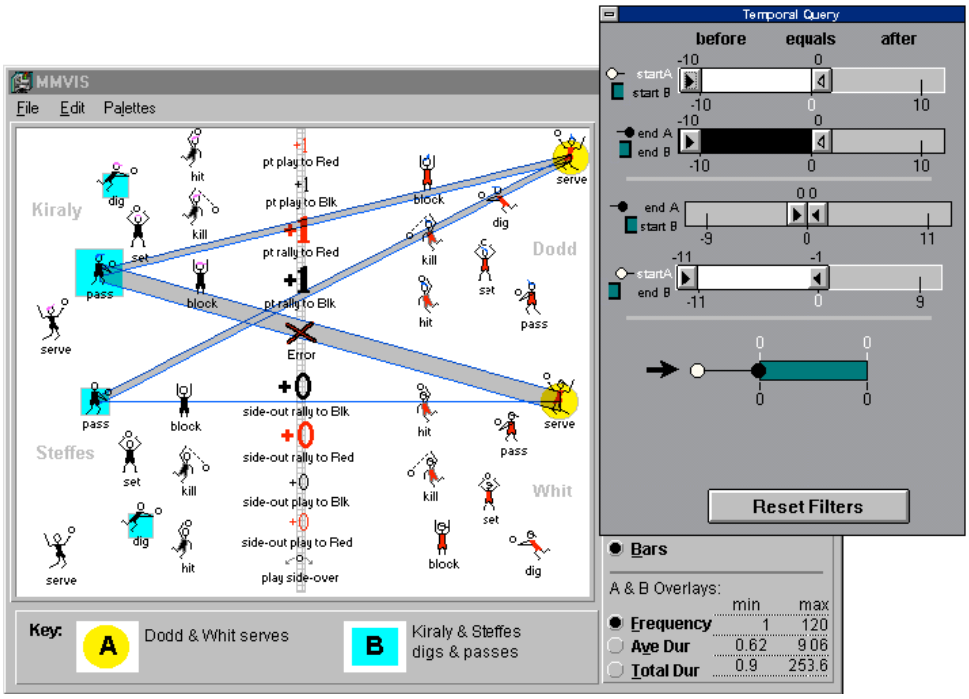


**Figure 3.42: MMVIS**

The figure also shows a temporal query – we are looking for instances where intervals in A are immediately followed by an interval belonging to B – in other words, where a serve from team red (Dodd and Whit) is followed by a dig or pass from team black (Kiraly and Steffes). The visualisation shows this as bars of varying thickness between the icons representing the players and their actions. The line from Whit to Kiraly is much thicker than the line to Steffes – this indicates that Kiraly receives far more of Whit's serves than Steffes does. On the other hand, Dodd's serves are equally distributed between Kiraly and Steffes.

An evaluation comparing this interface to a timeline-based analysis tool shows that it is less error-prone than the timeline tool, but that each had different strengths as to what kind of trends they were wont to discover. MMVIS has a rather specific mission, though; it is not very useful as a video retrieval tool, since it doesn't provide access to the video, or try to show what goes on when – it is only useful for analysing temporal trends (which it does admirably, however). Another nitpick may be that creating icons for the different actions and categories may be difficult and time-consuming, but nice icons is of course not a prerequisite for the tool being useful.

### 3.3.14 LifeLines

LifeLines [Plaisant et al. 1996] [Plaisant et al. 1998] has nothing to do with video, but it is nevertheless interesting in the present discourse. It is a visualisation environment for "personal histories", e.g. medical or court records. Facets of such a history are displayed on a common timeline; instantaneous events as icons and ongoing processes (medical conditions, for example) as lines. Line thickness and colour illustrate relationships or the significance of events. Figure 3.43 shows a LifeLines medical record.

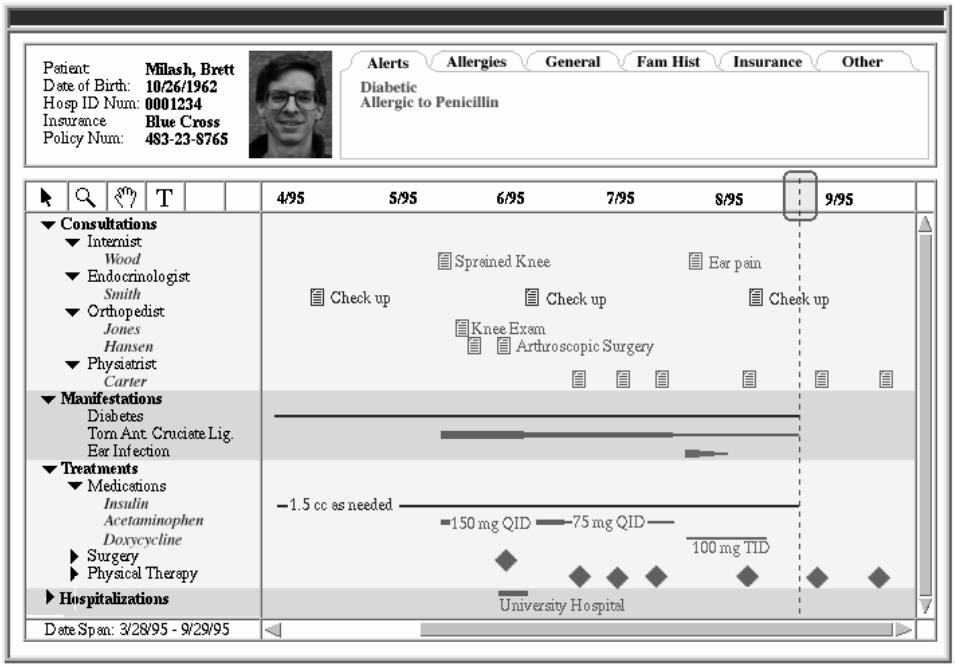


**Figure 3.43: LifeLines**

In this case, colour coding is used to connect related events – all the consultations, prescriptions and letters pertaining to a particular doctor or condition. Line thickness is used for severity of condition or amount of medication. Clicking on events brings up additional information about them in a separate window.

To fit more information on screen, labels can be removed and icons and lines packed more tightly. If the facets are organised hierarchically, LifeLines supports a summarisation feature, where a set of events are replaced by summary events when a facet is collapsed, as exemplified by the "Surgery" and "Physical Therapy" facets in Figure 3.43.

LifeLines is excellent at creating effective overviews of complex temporal data. Timeline displays are common among video annotation systems, but LifeLines presents some new ideas: using line thickness and colour to code additional aspects of the temporal data, and using hierarchical facets and summarisation to fit more information on screen. These techniques might be useful to look into in the context of video databases as well.

### 3.3.15 Discussion

The systems reviewed in this section have one thing in common: They are all very different. Their purposes and approaches include augmentation of paper notes, relevance ranking of free-text retrieval, interfaces for cooperative logging, metadata for language- and culture-independent retrieval, analysis of temporal trends and techniques for intra-video browsing. It seems hard to draw any conclusions about the state of the art in tools and interfaces for temporal annotations.

However, some things are worth mentioning. Most of these systems have a very narrow focus: They view the annotation data in one particular way, or they can only be used in one particular domain, for one particular purpose, or on one particular kind of video or audio data. Media Streams is the only one with a more ambitious outlook, but it may be argued that it perhaps goes too far in its complexity and richness – 44 strata and over 6000 icons can be overwhelming. Video databases with a more holistic approach do of course exist – Informedia [Christel et al. 1995] [Wactlar et al. 1996], for instance – but as mentioned in the introduction to this chapter, they do not have very interesting conceptual models.

Thus, it seems that a need similar to the one presented in section 3.2.12 can be identified: Creating an open and flexible, yet simple and user-friendly framework for producing, retrieving, browsing and analysing temporal annotations is a challenge worth looking into. There may possibly be great benefit in trying to integrate various information gathering techniques – browsing, searching, selecting analysing, zooming – instead of focusing on one or a few techniques in isolation.

One thing many of the mentioned systems do have in common is timelines. This is a relatively old and well-studied technique, with documented benefits. However, it is no panacea. It is a natural method for visualising temporal annotations, but user interface "laws" such as Shneiderman's mantra "Overview first, zoom and filter, then details on demand" [Shneiderman 1996] should be considered. Many of the systems have trouble with overviews – when an entire hour-long video is viewed at once, the displayed information along the timelines in LogJam, Jabber and Media Streams is in danger of being compressed beyond legibility. Rframes, the MSR Video Skimmer and the Hierarchical Video

Magnifier do not have that problem, but the information they present along the timeline is more syntactic than semantic, as is the speaker information in VoiceGraph.

Therefore, another question worth looking into is how better overviews may be created. Both temporal and "topical" overviews should be considered – topical overviews in the sense that a useful summary or aggregation of different kinds of annotations or strata should be available for the user. Of course, techniques for filtering and zooming should be considered in this context, according to Shneiderman's advice.

A third important observation is that though most systems offer video or audio browsing based on some non-temporal visualisation of the video, almost none presents more than one video at a time (with VoiceGraph as the sole exception). The other systems assume that the user first identifies the desired video document, probably through some query interface. This may be a false assumption. Query systems are difficult to use, especially in the case of semantic video content annotations, where the stored information is wont to be idiosyncratic and varying in completeness, level of detail and degree of structure.

An alternative is to investigate the possibility of browsing several video documents at the same time – to create an overview of an entire (portion of a) video database. This would enable the user to more easily get a feeling of "what is in there", and could also be useful for comparing similar videos, and analysing them for similarities and differences. This ties into the previous point made, about better overviews, and may have a significant impact on how a video database is accessed.

## 3.4 Summary

Video content models are plentiful, but lacking in flexibility and semantic expressiveness. The same goes for tools and user interfaces; many have interesting ideas, but are limited to particular purposes and access patterns.

In other words: A flexible, adaptive, and integrated framework for managing most, if not all, aspects of semantic content annotations, with powerful overview and inter-video browsing capabilities has not yet been created, and trying to do so has obvious merit.

# 4 Handling the semantics of video

This chapter presents more concrete rationales and requirements for handling the semantics of video in a computer system. Chapter 1 has touched upon this, but briefly and abstractly; chapter 2 has discussed the signs and semantics of video, but without putting it into the context of real-world problems to be solved; chapter 3 has presented the properties and shortcomings of existing systems, but not gone into detail on what could be done to improve them. In contrast, here I present actual problems I want to solve, discuss the specific issues they raise, and outline my concrete solutions to them.

Section 4.1 introduces six scenarios where semantic video annotations are useful or required, abstracted from real-world cases. Based on these, and on what I consider the shortcomings of the existing systems presented in chapter 3, a set of high-level requirements are identified in section 4.2. Section 4.3 introduces the main ideas I will build my approach on.

## 4.1 Examples of semantic annotation needs

The following scenarios are abstractions of real cases where computer support for video semantics is required. In this context, they serve several purposes. First of all, they may help a reader unfamiliar with such issues to understand better the need for semantic content annotations; to associate concrete problems and tasks to the more abstract purposes presented in section 1.1

Secondly, they also serve as a starting point for establishing requirements for annotation support, highlighting the shortcomings of expressiveness and tool support in existing systems. They are also useful for validating my designs later on; and will therefore be revisited in later chapters, some of them functioning as running examples and grounds for evaluation.

Thirdly, the selection of scenarios is intended to delimit the scope of the problem; to focus on the kind of video applications I consider to have the most need of and benefit from semantic annotation. Thus, they do not cover the entire breadth of video archive applications. For instance, I do not explicitly explore the requirements for managing the massive amounts of material that TV companies and film studios produce. Considering all possible application areas would be too big a task, so I concentrate on scenarios where the size of the material and the number of users are small to medium-sized, and where fairly detailed and purpose-specific annotation is desired.

Nevertheless, the scenarios are chosen so as to present a fairly manifold and extensive problem space. Most are from the knowledge work domain – documentation, research, analysis and education – as the need for content modelling is great here. However, it is not my intention that the system I propose should be restricted to this domain, or to the scenarios I present: I will strive to create a system flexible enough to tackle use cases I haven't thought of as well.

### 4.1.1 Support for movie watching

Film buff Jon enjoys Hong Kong gongfu[1] movies in general and martial arts actor Jackie Chan in particular. His problem is twofold: To find all the films that Chan has starred in, and to find the fight scenes within each film (since the films are typically rather inane plot-wise).

The first problem seems straightforward enough; one would think that any film database would be able to handle this. However, unbeknownst to Jon, Jackie Chan is sometimes credited as Yuen-Lung Chan, Long Cheng or Sing Lung, and has been the producer of several movies without starring in them. For the most complete and accurate results, Jon therefore needs a well-structured database, one where Jackie Chan is stored (along with his aliases) as an entity, not merely as a text string in the movie descriptions, and that distinguishes between different kind of person-movie relations (actor vs. producer).

Jon's second problem is partially solved by DVD menu systems. DVDs typically divide a movie into about twenty "chapters", and can on demand present a menu containing a keyframe from each chapter along with a chapter title. This may be sufficient for Jon to determine whether any fighting occurs, but each chapter is often long enough to contain several different scenes, and the keyframe and title may refer to one of the quieter scenes. It might be better if each chapter was described with keywords or a summary, or even better: a more or less formal classification of the type of action that occurs. But still, a chapter may have several minutes of non-action before the fighting begins, which may be a waste of Jon's time. This is a segmentation granularity problem. Using shorter chapters does not completely remove it, and introduces user interface problems regarding the presentation of and interaction with the larger number of chapters.

A better alternative is to identify the different types of action that occur in the film – fighting, car chases, romance, for instance – and associate them with temporal movie fragments independently of one another. The user interface could let the user pick an action type, and skip to the next relevant fragment. This avoids the granularity problem, and also enables people with interests different from Jon's to focus on other aspects of the movie – which is a good idea if the movie description is shared. If it is not, a user-defined segmentation will work just fine in Jon's case.

In summary, Jon's movie-finding requires a description model with fairly high semantic expressiveness. However, this is a non-temporal issue essentially similar to the problem of finding books in a library, so this thesis will focus more on his intra-movie browsing needs. As for them, Jon needs either a simple (but user-specified!) segmenting approach, or – if descriptions are to be shared – an annotation model based on free overlapping intervals. However, the need for semantic expressiveness is undemanding: simple keywords will suffice.

[1] Also spelled "kung fu".

### 4.1.2 Managing interview recordings

Cathrine works for the Norwegian Museum of Cultural History, and is creating an exhibition on clothing habits, fashion and self image among today's youth. As part of her research, she has recorded audio interviews with a selection of teenagers. Usually, such interviews are transcribed, but that is a horribly time-consuming and boring task. Instead, Cathrine digitises the recordings, and puts them in an audio database. Her main motivation is to be able to jump directly to a certain topic in an interview, and to quickly move to the same topic in different interviews, to review and compare them.

The interviews are fairly well-structured and similar, so Cathrine makes a simple list of topics (based on her interview design) and assigns them to non-overlapping segments of the different interviews. The segments corresponding to one particular topic become impractically long in some interviews, so she subdivides the topic into subtopics to keep the segments down to a manageable length. Analysing the interviews, Cathrine adds notes and comments to the various topics, and transcribes a few particularly illustrative quotes from the interviews. This material is later incorporated into the posters and presentations of the exhibition.

The main function of the annotation in this scenario is as a keyword index, but an additional requirement is that topics (keywords) and intervals can be described. User-defined segmentation is sufficient for temporal expressiveness, but it is interesting to note that a hierarchical organisation of topics is requested.

### 4.1.3 Support for system analysis

Ian is a project leader at the Norwegian Defence Research Establishment, in charge of developing a new command and control information (CCI) system for the Norwegian military. To analyse the current system and help establish requirements for the new, he has recorded tens of hours of video from military staff meetings, exercises and manoeuvres. These videos are to be used by his team for two related, but different purposes: to analyse the practises and problem areas of the existing system, and to establish requirements for the new.

First, Ian decides to coarsely index the video with some simple information; what is going on, who is doing something (rank, role and responsibility being more important than person identity), where it is taking place (the role of the place, e.g. staff headquarters, is likewise more important than the actual, physical location) and when it is happening. Using this index, the team members can quickly find the parts of the videos that they want to analyse more closely.

During this analysis, an ontology of people, activities, roles, places and procedures is gradually established. Activities are classified and divided into subactivities; places are related along communication lines; roles are abstracted away from people, classified and organised, and related to the activities they perform. These entities are connected to intervals in the videos, acting as examples, illustrations and a link to the real world. Entities and intervals are also connected to Ian's web-based requirements database through URLs.

This scenario illustrates the use of ontologies of high semantic expressiveness for video annotation: A complex model of the real world is constructed and linked to relevant video material. The ontology captures the knowledge produced by the system analysis, and grounds it in the real world through the association with video intervals. Thus, the annotation develops into more than just a video index, and the video connection may even become less important as the project proceeds. Another important observation in this case is that the ontology (and its video association) is constructed incrementally, as the understanding of the systems and processes depicted in the video develops.

### 4.1.4 Video-based system evaluation

Hallvard is a researcher evaluating the effectiveness of electronic medical records (EMR). He videotapes patient-doctor consultations, and analyses the video to determine how time is spent, and how the physician uses the various electronic tools and databases at his disposal.

Hallvard needs to describe intervals in the video with two main schemes. Firstly, he wants to partition the consultation into segments or phases – preliminary work, introduction, anamnesis, examination and so on. These phases should not overlap. Secondly, the various activities performed by the physician should be indicated. These activities are hierarchically organised; the high-level activity "Acquiring information" may be subdivided into "Accessing catalogue", "Reading papers" and "Using PC". Activities may also overlap – the doctor may very well take notes while questioning the patient.

After annotating the video with these phase and activity indicators, Hallvard wishes to compute statistics based on them. He wants to compute the total time used by every activity, relative to the length of the consultation, both on the lowest level and on the higher levels of the activity hierarchy. He also wants to create time expenditure statistics within each phase of the consultations – e.g. to establish in which phases of the consultation the "Documenting" activity is used the most. Finally, he also wants a spatial, configurable visualisation of the intervals and the ontology, as he believes this may help him discover trends and patterns in how the activities of a medical consultation is influenced by the use of EMR.

### 4.1.5 Lecture database

Steinar is the lecturer of an elementary computer science and programming course at the Norwegian University of Science and Technology. As a service to his students, he records his lectures – that is, the screen activity on his PC (mainly PowerPoint presentations and demonstrations of programming) along with his aural exposition – and publishes them on the university intranet.

For the whole course, this amounts to over forty hours of video, so Steinar wants to provide a topic-based searching and browsing interface. He wants to assign topics to intervals of the video, and organise the topics in a hierarchy. For instance, he would like to be able to assign a fifteen-minute segment of a lecture to the topic "while loops", which is a subtopic of "loops", which is a subtopic of

"flow control", which is a subtopic of "programming with Java Server Pages". Students searching for or browsing "flow control" should come up with the while loops segment, though it's only indirectly connected through the subtopic relation.

Steinar also wants to classify segments as exposition/explanation or demonstration/example. This classification will overlap with the topic classification – the treatment of the while loop will probably start with exposition, followed by one or more examples. Topics may also overlap; the topic "conditions" (a subtopic of "expressions", subtopic of "programming with Java Server Pages") will probably be revisited during the while loop explanation.

Last but not least, Steinar would like to attach additional information to the segments and topics. This may be references to the course literature (when it isn't present in the video), corrections of errors, better explanations of difficult bits, or links to files containing example code. He also toys with the idea of letting the students enter their own comments and notes into the system.

### 4.1.6 Police investigation

Jostein is a police officer investigating a homicide. The crime was committed in Geiranger, one of Norway's most popular areas of natural beauty, so Jostein has several hundred hours of tourist videos taken around the time and place of the crime. Jostein hopes that these videos may help to establish the movements of people and vehicles in the period, and thus help solve the case.

To do this, Jostein needs to record the presence of different people and vehicles at different times and locations in the video. Positive identification of men and cars is difficult and uncertain, but Jostein and his co-workers enter detailed descriptions of them – height, build, hair, clothes and so on for people; make, model, colour, license plates etc. for cars – and records which intervals in which video they occur in. Subsequently, they use a search system to look for similar objects, and use manual inspection of the video to ascertain if they are the same or not. In this way, patterns of movements may be established, and suspicious vehicles or persons brought to attention. A detailed ontology of the geography of the Geiranger area is also developed, to better describe the location of different entities.

A related important task is to establish the timelines of each video. This is essential to determine and extrapolate the movements and whereabouts of entities filmed in different videos. Jostein and his colleagues segment the video material into continuous shots, and record the real time at which the recording was made for each shot, where it is known. Depending on camera capabilities and setup, the time information may not be readily available, but guesses can be made, and comparison with videos with more reliable timelines may reduce the uncertainty.

Like Ian's system analysis scenario, this one also builds a detailed model of reality based on the video material. What makes this one more demanding is the stronger importance, but also uncertainty, of time, place and identity. Semantic

expressiveness powerful enough to express belief, possibility and probability is needed.

## 4.2 Requirements for a multi-purpose, flexible video content model

In a sense, the entire thesis up to this point has been concerned with establishing requirements for a sensible video annotation system/framework. From the abstract visions of chapter 1 and the study of film semantics in chapter 2, through the survey of the accomplishments and shortcomings of existing systems in chapter 3 to the scenarios in the previous section, the aim has been to explore

- what is needed for useful semantic video annotation, and
- what needs to be done to reach that goal.

This section crystallises my position in this regard: It presents the requirements I will choose to focus on, to what degree they are supported by existing systems, and what needs to be done to "fill in the holes" in the state of the art, so to speak.

### 4.2.1 Temporal expressiveness

The scenarios show (unsurprisingly) that some temporal subdivision of the video material is needed, but the exact requirements vary. Jon, for instance, needs to divide each film into scenes (using the term loosely), in order to classify them coarsely according to the type of action in each scene – fighting, chases, romance, confrontations etc. Some scenes may overlap; e.g. is it quite possible that a confrontation scene includes a fighting scene. Cathrine has a similar requirement, but she is content with a simple partitioning of her material. Hallvard and Steinar need to describe the video with activities, topics and processes that are relatively independent of one another, thus needing a fairly complex and expressive temporal subdivision. Ian and Jostein may also need this, but it is also possible that a simpler segmentation scheme is sufficient for their purposes. All in all, the scenarios cover the spectrum of temporal expressiveness, from lowest to highest.

Hence, it is my position that the model should aim for *full temporal expressiveness*. Segmented approaches are simply too restrictive, as they do not easily support the modelling of different, independent aspects of the video material.

There are certainly advantages to segmentation. It is a simpler principle for the end users to grasp, and structural features of video, e.g. its subdivision into shots, correspond well to segmentation. A fully temporally expressive model can handle it equally well, as a special case, but tools and user interfaces geared towards handling stratified, independently overlapping intervals will probably be awkward if used to work with segments.

It is possible to provide support for both segmented and full temporal expressiveness, as VideoSTAR [Hjelsvold 1995] does. However, it is my position that this increases the complexity of the model too much for too little gain, violating my requirements on simplicity and usability as described later in

subsection 4.2.5. I believe stratified annotations should be prioritized over segmented, due to their greater expressiveness. This work is focused on *semantic* annotations; representing *structural* features, which is the segmented approach's forte, is demoted in importance.

The existing systems in section 3.1 cover the breadth of temporal expressiveness, so this requirement does not entail much in the way of novel research. The organisation of the intervals (and/or the semantic annotations tied to them) is another matter, but this will be covered in subsection 4.2.3.

### 4.2.2 Spatial expressiveness

The scenarios do not mention spatial annotation explicitly; indeed, it is quite clear that neither Jon, Cathrine, Hallvard nor Steinar require it. A case might be made for Ian and Jostein, though: Jostein in particular might need to be able to indicate which is which when two persons or vehicles appear simultaneously in a video interval. (Please note that the ability to describe places seen in the video does not constitute spatial annotation in my usage of the term.)

Nevertheless, I choose for the time being to ignore spatial expressiveness. The need is in my opinion not great enough to justify the unavoidable complexity a spatial annotation scheme would entail. Jostein will most likely manage by using properties to distinguish entities – if the annotations record that a red car and a blue car is present in some video interval, visual inspection should be sufficient to tell which is which. It is interesting to note that of all the annotation models described in chapter 3, only BilVideo [Dönderler et al. 2003] provides any spatial expressiveness.

### 4.2.3 Semantic expressiveness and flexibility

Given a temporal annotation scheme (and the lack of a spatial one), the next issue is to decide how each time-delimited piece of video should be described. Jon, for instance, needs a simple classification. This might very well be specific to each film, as he needs intra-film navigation only. Additionally, the ability to assign a title or description to each scene would be useful. Cathrine, on the other hand, requires a common topic list for all her interviews; what is more, she wants to organise the topics in a hierarchy. She also needs to describe the topics as well as the media intervals. Hallvard and Steinar have similar requirements. Steinar in particular needs a hierarchy with arbitrary depth, and also subsumption semantics for the hierarchy – in other words, the organisation of the topics must be more than just cosmetic. Ian and Jostein have the most complex requirements in this regard: They need complex representations of real-world objects, places and activities, described by properties and connected to each other with different kinds of relationships.

This places demands on both the semantic expressiveness and the flexibility of the model. To handle the most advanced scenarios, the highest levels of semantic expressiveness and flexibility, as defined in section 3.1, is needed. This entails complexity, but I consider this to be an important requirement. To be concrete, the model must support annotation approaches as diverse as

- a simple list of topics
- a hierarchy of topics, with descriptions
- as above, but including defined semantics for the hierarchy relation
- an arbitrarily complex semantic network of concepts with properties

The challenge is to combine this power with simplicity and usability. The model should be basically simple, but *able to represent complex knowledge precisely*, if needed. A simple basic structure is good for usability; it makes learning the system an easier and less daunting task for the user, and it is sufficient for simple applications. The model should be *usable for multiple purposes*, including complex ones, so it must be easily *extensible*, and capable of handling arbitrary objects, properties and relationships between them, including generalisation/specialisation and class/instance relationships. Moreover, it should be *changeable on the fly by the end user* – a computer scientist or engineer should not be needed. The user should be able to gradually change, extend and improve his model as his needs change and his understanding of the domain evolves.

The model should employ *concept-based annotation* – that is, relate each temporal interval to a concept, an identifiable object of interest in the universe of discourse, as mentioned in section 3.1.1. This way of organising descriptions (in contrast to just describing intervals with simple properties) may be a slight obstacle for the novice user, but it has very many advantages, as noted by [Weinstein 1998]:

- It factors out the content-independent bits of information (like a person's name), avoiding redundancy.
- Concepts can be reused wherever they are applicable, saving time, increasing preciseness and avoiding ambiguity and spelling problems.
- It allows for easy and exact analysis of the usages of each concept.
- It facilitates browsing and searching, as the set of concepts can be used as an index or catalogue.

Section 3.1.1 mentioned a potential problem with this approach, though: At what level of detail should the concepts be constructed? With full temporal expressiveness, the problems of temporal granularity associated with segmented schemes are avoided, but a similar problem of *topical granularity* crops up here. When annotating (say) anchorpersons in a news broadcast, several degrees of topical granularity are conceivable. Some might find it sufficient to record when an anchorperson (any of them) is speaking or not, and thus use a single "anchorperson" concept. Other might be interested in whether a male or female anchorperson is speaking – in that case two more specific concepts, "anchorman" and "anchorwoman" is needed. Others again might need to know the identity of the person in each instance, and require a concept for each occurring person. The model should take this into consideration. For instance, it would be useful if it were possible to *hide unnecessary details* from users who do not need it – to automatically e.g. aggregate the most specific anchorperson annotation described

above into the simplest simple "anchorperson speaking or not" scheme. Likewise, the model should facilitate the addition (or removal) of details and more specific concepts incrementally, as the user's requirements change. This is dependent on the tools and user interfaces as well as the model itself.

This is perhaps the most demanding aspect of my requirements. The powerful expressiveness and flexibility is one thing; combined with the simplicity and usability required for it to be a feasible solution for the simple scenarios, it is a real challenge.

The state of the art in this regard is in my opinion lacking. Granted, Jon and Cathrine have their requirements fulfilled by almost all of the models in section 3.1. Some are less than perfect – Veggie does not support the overlapping temporal intervals that Jon might need, and Cathrine's hierarchical topics may be cumbersome to represent in some models – but none are useless. For Hallvard and Steinar, some models are found wanting. The need for free, overlapping temporal intervals and user-specified, hierarchical topics or strata precludes the use of half of them, and the rest are not perfectly suited, either. For the most advanced scenarios, Ian and Jostein, none of the models are suitable. Smart VideoText and BilVideo have flexibility and semantic expressiveness, but the one is geared towards formal understanding of prose descriptions and the other towards spatiotemporal objects, neither of which is a perfect fit. VideoSTAR might not be so bad for Ian, since its world view (like his) consists of people, locations and events, but Ian needs to classify, describe and relate these in greater detail than VideoSTAR supports.

It is also my goal that a single model or framework should cater for all the needs of all the scenarios. Although the most powerful of the existing systems fulfil a significant part of these requirements, the complexity of their models and their reliance on the (in my opinion) relatively esoteric technologies of conceptual graphs and logic programming make them unsuitable for the less demanding scenarios.

### 4.2.4 Tools, interfaces and visualisation

In the scenarios, different kinds of tools and user interfaces are required for using, manipulating and displaying the annotations. Most, if not all of the scenarios require browsing and navigation capabilities based on the temporal subdivision of the material and the topic structure. Some require no more, but others are more demanding: Hallvard wants statistical analysis on the length and frequency of his topics, as well as a configurable, spatial visualisation of the annotations in order to look for patterns; Steinar wants a search system that is aware of the semantics of his topic hierarchy; and Jostein wants software support for finding similarities between object descriptions.

These tool support requirements are rather varied, and section 3.3 also illustrates how many wildly different kinds of tools may be built for semantic annotation support. There are too many possibilities for me to cover them all, so I will focus on one purpose in particular: visualisation and browsing.

Simple visualisations, typically based on time lines, are used for media navigation and other interaction in many systems. However, in chapter 3 I argue that the state of the art is limited in this respect: Visualisation and browsing almost invariably handles just one video at the time; query languages are used to select a video for browsing. I argue that query languages have limited user-friendliness, and that they are poorly suited to the idiosyncrasies of semantic video annotation. Existing visualisation techniques are limited; many are unable to present a usable overview of even a single video. Time-line-based visualisation and browsing at *all* levels of detail is a better, but untried, idea.

Hence, I require *inter-video visualisation and browsing*. The model and the tools and user interfaces should support *visual overviews* of the entire video database content, let the user *zoom in* on selected parts and *filter out* irrelevant information, and provide *details on demand*, according to Shneiderman's mantra [Shneiderman 1996]. This dovetails neatly with the discussion on hierarchical topic structures and topical granularity in the previous subsection.

### 4.2.5 Simplicity and usability

The perhaps biggest challenge in the design of such a system is to support the requirements of the most complex scenarios without being too complex for the less demanding users. Hence (among other reasons), I choose to ignore spatial annotations, and decide against supporting both free intervals and a segmented scheme for temporal expressiveness. However, the semantic expressiveness and flexibility I require still makes this a challenge.

Simplicity and usability are non-functional requirements, hard to quantify. However, I will require that the basic conceptual model for my system should not be overly complex – to be concrete, it should be possible to depict it with less than ten boxes in a diagram. As for usability, users such as Jon and Cathrine should be able to construct simple topic lists and annotate intervals and topics with text descriptions without having to struggle with type systems and property definition. Tools and interfaces for retrieval/browsing (i.e. where the annotations are not edited) should be usable without more instructions than might fit on a single web page.

### 4.2.6 Summary

In short, these are the requirements and goals for my proposed semantic video annotation system:

- Full temporal expressiveness
- No spatial expressiveness
- Concept-based annotation, with the ability to classify concepts, arrange them hierarchically with defined semantics, and define properties and property values, while keeping complexity to a minimum
- Intra-video, semantics-aware visualisation and browsing, including overviews, zooming (both temporal and topical) and filtering (ditto).

- A model and tools simple enough for an undemanding user to grasp the model structure and annotate a video with a simple topic list and textual descriptions without hassle.

## 4.3 The road ahead

The ideas I wish to explore further as a contribution to the field of video databases have been lurking in the background in the previous chapters; they also pervade the scenarios and requirements presented in the sections above. It is time to crystallise exactly what I mean. This section presents, in a concise manner, my two main points: ontology-based stratified annotation, and exploitation of ontology structure and semantics in the visualisation of annotations. It also reviews the research questions from section 1.2, relating them to the current discussion and the organisation of the rest of the thesis.

### 4.3.1 Ontology-based annotation

Full temporal expressiveness and concept-based annotation is in my opinion a necessary requirement. The most significant question remaining is *how to define, organise and manage the concepts*. This has implications for almost all aspects of the model – its flexibility, semantic expressiveness and usability. It also has great impact on the problem of topical granularity, which I wish to present a solution to.

Considering the scenarios, both Cathrine and Steinar require a hierarchy of topics. This suggests that a hierarchical organisation of concepts would be a good idea. Indeed, film semiologist Christian Metz [Metz 1974] presents the *codes* (signs and syntax) of the film "language" as a hierarchy. Steinar additionally wants subsumption semantics – a set of video intervals relevant to a given concept (topic) is implicitly also considered relevant to the concept's ancestors.

Thus, organising the concepts in a hierarchy with subsumption semantics is a good starting point. This also covers the simpler cases where just a list of concepts is required – a list may be considered a degenerate hierarchy. However, it is useful to consider this hierarchical organisation closer, to see if further advantages can be gained without too much complication or effort.

Ian and Jostein need the model to handle user-defined objects and properties. For this to be done formally, a *type system* – a mechanism for defining what kinds of things we may talk about – is required. A system in the object-oriented style, with classes, subclasses and instances [Budd 1998] seems an ideal fit: It is powerful and convenient, and specifies a hierarchy of entities (or a directed graph in the general case) with subsumption semantics. Hence, it can be used also in the simpler cases.

Jostein and Ian might also want to specify user-defined relationships between their concepts; to specify *kinds* of relationships, and make inferences about them. Relationships are important in the analysis of video; as discussed in section 2.2.2, two of the three kinds of signs in the video "language" (Indexes and Symbols) are based on relationships that must be learned in order for the signs to be understood

correctly. If such understanding, interpretation and inference is to be supported by a computer system, it is necessary to use formal *ontologies* (see appendix A). Ontology frameworks typically include an object-oriented style type system; thus, organising the concepts of a concept-based annotation scheme in an ontology facilitates both the most complex and the simplest requirements for semantic content annotation of video.

This constitutes an *ontology-based stratified annotation scheme*. A stratum is the set of video intervals connected to a given concept, and the concepts (classes and objects) are formally defined and described in an ontology. This gives several advantages:

- It allows the user to define the "universe of discourse" using formal, powerful constructs, i.e. ontology languages, to an arbitrary level of detail. It allows a heterogeneous yet well-defined and well-structured description of the various "things" or topics relevant to the video(s) in question.
- It imposes a structure on the annotation strata, through their connection to the corresponding concepts. Instead of a flat list of strata, we now have a hierarchy or partial order with defined semantics.
- It defines relationships between the concepts, that can be used in interaction with the system, e.g. for query expansion/contraction, relevance ranking, query disambiguation, browsing and visualisation.

In short, such an organisation provides the means for very detailed and precise semantic annotations, while still being simple enough not to overwhelm users with less complicated demands. The "universe of discourse" may be as simple as a plain list of topics, or as complex as desired, all within the same framework.

The power of ontologies in organising concepts (and annotation strata) can be used for many different purposes, as mentioned above. Some of these will be explored further in the remainder of this thesis; others will be deferred to future research (see section 9.2). One application I have chosen to focus particularly on is visualisation and browsing.

### 4.3.2 Video browsing using aggregated visualisations

As discussed in the previous chapter, it is my position that existing work on video content management has been too focused on query languages, neglecting the possible advantages of visualisation and browsing systems. In particular, visual overviews and facilities for inter-video browsing are areas I wish to improve.

Stratified timelines, as used by e.g. [Davis 1993], [Kazman et al. 1996], [Carrer et al. 1997] and [Cohen et al. 1999] is a tried and true paradigm. Lines, rectangles or arrows represent temporal intervals where a particular concept or stratum is relevant; and the strata are stacked on top of each other (see e.g. Figure 3.12, Figure 3.36 and Figure 3.40). However, the number of strata that can be fitted on screen is limited, typically in the order of 20-30. This may not be sufficient to view all the strata of a single video, let alone a collection of videos. Additionally,

all the strata may not be of interest to the user, or they may be at an inappropriate level of detail.

My suggested solution to these problems is closely related to the idea of ontology-based stratification: The concepts/strata are presented as a collapsible tree list (like the folders in Microsoft Windows Explorer), a common and easy-to-use user interface widget. Using this, the user can control which strata are displayed by collapsing and expanding nodes, hiding uninteresting strata and fitting more relevant information on the screen. Figure 4.1 shows an example of an annotated news broadcast. The Anchorperson concept has two subconcepts – Anchorman and Anchorwoman – that again have subconcepts representing individual anchors, but the user is not interested in that level of detail, and so leaves the Anchorperson node unexpanded. The other concepts are treated similarly.

Utilising the subsumption semantics of the hierarchy of concepts, it is possible to construct *aggregated visualisations* for the collapsed concepts; that is, incorporate the timeline representation of its descendants into its own representation in some manner. Instead of (or in addition to) the plain lines, rectangles or arrows normally used in timeline displays, graphical hints such as line thickness, colour or patterns can be used to indicate information aggregated by the collapsed concept/stratum. This is also shown in Figure 4.1. The annotations shown are intervals related to hidden subconcepts, e.g. individual anchorpersons. Since each individual anchor is an Anchorperson, their intervals are deemed relevant to the Anchorperson concept, and shown even though the node is collapsed.

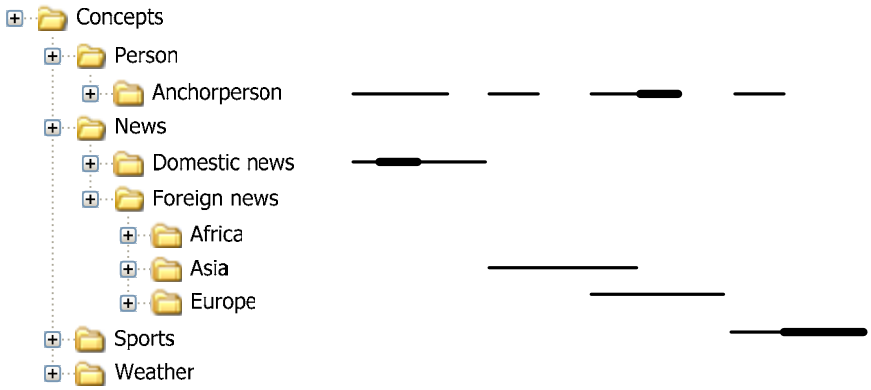


**Figure 4.1: Simple mock-up of visualisation and browsing interface**

The thicker interval lines are examples of how such aggregation may be visualised when the intervals of several subconcepts overlap, as the Asia and Europe concepts do. For instance, the thicker line segment for the "Domestic news" concept indicates that two concepts, e.g. "Election" and "Political demonstration", overlap in the corresponding time interval. While it is not possible to identify which concepts it is without expanding the node (or using mouseover text, tool tips or a similar technique), this is useful to indicate the amount of annotation and to create a visual overview or signature of the video content.

With this kind of visualisation and interaction, the user may adjust the level of detail of the timeline display merely by manipulating an ordinary tree list, alleviating the topical granularity problem. In fact, as the expanding and collapsing of concept nodes functions as a "topical zoom", and makes it feasible to present several videos on the same timeline, allowing the users to "drill down / zoom in" on the strata they finds most interesting.

### 4.3.3 Research questions revisited

In chapter 1, three questions were posed as the overarching problem statement of this research. At this stage – the logical as well as physical midpoint of the thesis – it is natural to review these questions, in light of the background material presented up to this point, and the new contributions to follow.

**Video information modelling**

*How should a model for semantic, temporal annotations be constructed, given that it should be usable for quite different domains, purposes and levels of detail?*

- What are the requirements concerning such a model? What kinds of semantics, what level of expressiveness should be supported?
- How can the model be made flexible or extensible enough to support different kinds of applications, while not sacrificing simplicity and usability?

The first part of this question has been addressed. Based on deficiencies and shortcomings (as I see them) in existing systems, my opinions on what kind of applications have the greatest need for semantic video annotation, and a set of scenarios based on real-life cases, I have presented my requirements concerning a model for semantic, temporal annotations.

The second topic has been broached: I have submitted that using ontologies for stratified annotation may be a solution to the problem. I have not yet gone into detail, but I will do so in chapter 5, which begins with a concrete conceptual model for ontology-based video annotation.

**Tools and interfaces**

*How can one create an infrastructure to handle the various tasks related to the usage of semantic content annotations?*

- What kinds of architectures and technologies can or should be used? How should the metadata be stored, accessed and transferred to the user?
- How should tools for entering, browsing, presenting and querying annotations be constructed?

The infrastructure has not been touched upon so much yet. Architectures and technologies will also be discussed in chapter 5, after the presentation of the model and a preliminary discussion on its properties.

However, I have given presentation, browsing and querying some thought: Based on the same background as the model requirements, I have hypothesised that inter-video, semantics-aware visualisation and browsing is an interesting and fairly novel approach to building tools and user interfaces, which also meshes well with ontology-based stratification. Hence, I will prioritise exploring these possibilities; if necessary at the expense of the storage issues also mentioned in the question.

The concrete objectives, design and implementation of tools and user interfaces are presented in chapter 6.

**Usability and performance**

*How does such a system fare in practice, in the real world?*

- Does it provide real benefits compared to prior approaches found in literature? Under what circumstances is it appropriate and successful, and when is it not? Is there an actual need for this kind of system?
- Are the model and the tools user-friendly enough? Expressive enough? Extensible enough? Simple enough?
- Is the model possible to index and search efficiently? Is it scalable? How does the choice of architecture/technology affect this?

The last question has obviously not been discussed yet; the design and implementation of a system must come before the evaluation. It may nevertheless be useful to give it some consideration at this time.

Part of the answer to the first bullet point is in a sense already given. My designs will focus on issues not satisfactorily solved in existing system; that was one of my criteria for determining the requirements presented in this chapter. Likewise, the scenarios in section 4.1 indicate circumstances where I believe my designs may be successful and needed. Whether these assumptions and decisions are correct remains to be seen, though.

To ascertain this, and to evaluate the system under realistic circumstances, I will seek to try out several of the scenarios in practise. Another important task is to evaluate the efficacy of my projected inter-video, semantics-aware visualisation and browsing interface, compared to more traditional approaches.

The evaluation of the OntoLog system is described in chapter 7. The results are discussed in chapter 8, and chapter 9 concludes the thesis.

# 5 OntoLog: a flexible video content model

The objective of this chapter is to present and explain the OntoLog annotation model. In section 5.1 I describe the basic conceptual structure of the model; its elements and semantics. This is followed in section 0 by a preliminary discussion on the power of the model: How its features can be utilised in practice, and some of its potential issues, based on the scenarios from section 4.1. Section 5.3 presents technologies relevant to the implementation of an OntoLog system, and the subsequent section discusses my technology choices. Section 5.5 summarises the chapter.

## 5.1 The OntoLog model

This section presents the OntoLog video content model in several steps. Its most important feature is how video intervals are organised into concept-based description strata, and how the concepts are organised. Thus, the first two parts of this section present the "core" model: how videos, temporal intervals and concepts are defined and interact. Subsection 5.1.3 explains the mechanisms for customisation and extension that this model affords, while the fourth and last subsection presents additional "convenience" entities and relationships that are not crucial for the OntoLog modelling approach, but are useful in an actual implementation nonetheless.

A conceptual UML diagram [Fowler 2003] of the core of the OntoLog video content model is shown in Figure 5.1. This is deliberately a very simple diagram, designed to present the "bare bones" of the OntoLog approach, its most crucial parts. The expressiveness of the model is greater than a cursory reading of this diagram might suggest, as will hopefully become clear in the following discussion.

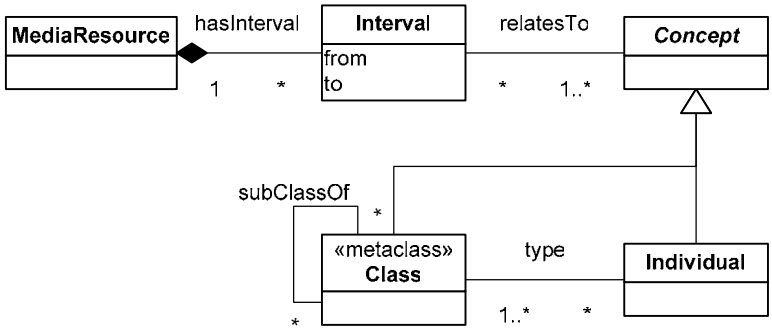


**Figure 5.1: An UML view of OntoLog's core conceptual model**

### 5.1.1 MediaResources and Intervals

The *MediaResource* element represents the digital media objects, e.g. MPEG files. Each MediaResource contains an unbounded number of *Intervals*, with start time and end time. By default, no restrictions are put upon the temporal ordering of the intervals, so they may freely overlap. Both MediaResources and Intervals

may be described further – each MediaResource ought to have a name or title, for instance – but the mechanism for this is described later, in subsection 5.1.3.

### 5.1.2 Ontology-based stratification

There are two principal annotation methods in the OntoLog model. The main mechanism is based on each Interval being connected (related) to one or more *Concepts*. There are two kinds of Concepts[1], *classes* and *individuals*, related to each other with specific semantics. Concepts may represent terms, topics, persons, places, events – anything that it is desirable to mark the presence of in the media object.

#### Classes and individuals

Discussions about individuals (or *objects* or *instances*, as they are often called) and classes abound in literature about object-orientation and ontologies, but there is no consensus on what precisely the terms mean [Date 1996]. Hence, it may be useful to present the definitions that OntoLog uses.

A *class* is a set (possibly unbounded) of values; as such, it is not much different from what is normally called a *data type*. However, unlike data types, the OntoLog classes do not place any restrictions on the nature or representation of its values; a value is a member of a class simply by virtue of explicitly stating that it is. It follows that a value in general may be a member of multiple classes.

"Value" is a very general term. I therefore use the term *individual* for the user-defined values belonging to the user-defined classes used to annotate video in the OntoLog model. The "type" association between class and individual in Figure 5.1 is the mechanism for individuals to state what class(es) they belong to. The UML diagram is thus slightly imprecise; there is no "Individual" class. The box represents all possible user-defined individuals, belonging to user-defined classes. Likewise, there is no "Concept" class either; it is merely an abstraction to signify that there are two kinds of concepts, and that Intervals may be related to either (or both) kinds.

Classes being sets, the "subClassOf" association is a subset relationship. It follows that members of a class are also members of its superclasses, and that a class may be a subclass of more than one class.

Note that despite this looking very much like an object-oriented type system, operators (or methods) do not come into the picture at all.

#### Ontologies and strata

A set of related classes and individuals (and properties describing them) constitutes an *ontology*. Admittedly, this is a very simplistic ontology framework, but classes and individuals constitute the common denominator of all ontology

[1] Note that some definitions of ontologies uses "concept" as a synonym for "class"; other uses it as a synonym for "instance". I use it as a hypernym for both classes and members (instances) of classes.

system, as described in appendix A. Thus, this represents the bare minimum for an ontology-based annotation model. Facilities for more complex ontologies can be provided by existing ontology languages (presented later in this chapter); there is little need for me to create a custom scheme for the OntoLog model, or to clutter this introductory presentation of the model with a more complex scheme.

Concepts may be thought of as organised into hierarchies. Classes are subclasses of other classes, and individuals are instances of classes. In the general case, this is a directed graph (due to the multiple subclassing and multiple typing discussed above), but in practise I assume most class graphs will be quite hierarchical.

Intervals and Concepts together create a stratified annotations scheme, where the concepts act as strata. It also creates a *hierarchical organisation of the strata*, which is an important point. Using concepts or categories for logging video typically produces a lot of categories – the experimental project described in [Cohen et al. 1999] used about 80. A flat list of this size is quite unwieldy, but arranged in a hierarchy, it is far easier to use. It also allows for easy aggregation of annotations and customisable level of detail during both logging and browsing, due to the subset semantics of the hierarchical relationships: The intervals relevant to a concept are not only the ones directly connected to it, but also the intervals connected to its descendants, i.e. to the transitive closure of the subclass-of and individual-of relationships.

Consider the annotation of a news video: One would create a Person class, with subclasses Anchor and Reporter, and enter actual anchorpersons and reporters as individuals of these subclasses. Figure 5.2 illustrates this example. The video would then be annotated by recording the presence of the different people in different Intervals. Now, if another user is interested in the presence or absence of anchors, but doesn't care about the actual identity of them, it is a simple matter for a computer system to use the hierarchical relations to aggregate the intervals related to the anchors, and present them as though directly related to the Anchor class. Thus, unnecessary information is hidden, and a customisable level of detail may easily be achieved. The user may be presented with concise, aggregated information as the default, and may delve into the parts of the ontology she is most interested in. Similarly, an annotator not concerned with the identity of people may relate Intervals directly to the Anchor and Reporter (or even Person) classes. This makes the annotations less useful for a user in need of more details, but is useful if e.g. the identity of the person is unknown, his or her appearance rare enough not to warrant the specification of an individual, or if the ontology is developed incrementally during annotation.

Figure 5.2 also illustrates another point: that classes are values belonging to a class (through the "type" relationship) as well. This makes for a homogeneous and simple model; there is no semantic or logical difference between system-defined and user-defined classes. It does however blur the distinction between class and individual, but I will remedy this by defining a class as a set-value belonging to the (system-defined) Class metaclass, and an individual as a value belonging to a class, but *not* to the (system-defined) Class metaclass. Thus, Tom, Dick and Harry are individuals; and Person, Anchor and Reporter are classes.

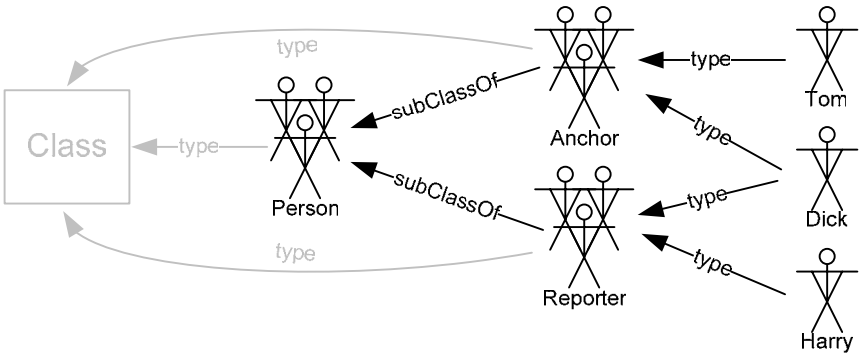


**Figure 5.2: Classes and individuals**

### Intervals with multiple Concepts

Intervals may be related to more than one Concept. Consider a scientist studying the social habits of chimpanzees, by recording them on video and analysing the footage using the OntoLog model. She would probably want to record different activities – chest thumping, grooming, playing – but would also be interested in recording which chimp is performing the activity. This could be done in different ways: she could use activities as Concepts, and record the active chimp as a property to each Interval. This, however, might make it somewhat difficult to follow the activities of a particular chimp, compared to if she used the individual apes as Concepts. But if she did that, and used activities as Interval properties, the opposite problem would occur. She could use both individual chimpanzees and activities as Concepts, but that would in some cases lead to ambiguity: When several chimps are doing several activities at once, which is doing what?

For that reason, an Interval may be related to more than one Concept. Thus, the scientist could record the performance of an activity by an ape as a single Interval, related to a chimp Concept as well as an activity Concept, avoiding ambiguity and enabling strata-based visualisation for both ape-oriented and activity-oriented views of the situation.

### The semantics of "relatesTo"

The discussion above is actually a simplification: Consider the case where an activity involves more than one chimpanzee, e.g. grooming, which often involves both a groomer and a groomee. You can connect the interval in question to both the "Grooming" concept and to the two chimpanzee concepts, but the groomer/groomee information is lost.

There are several possible solutions to this problem. One is to consider "Grooming" a class, define a Grooming individual for each Interval, and describe this individual with "groomer" and "groomee" properties, the values of which are chimpanzee individuals. This captures the information adequately, but has the same problem as earlier: It is impractical to follow the activities of one particular ape; the stratification is no longer both activity- and ape-based. It also leads to a lot of Grooming individuals that are "used" only once.

Another solution is based on the observation that the "relatesTo" association is a very coarse – perhaps too coarse? – indicator of the relationship between an Interval and a Concept. It may signify fairly different things:

- that an individual is seen in the picture during the Interval
- that some (unidentified) member of a class is seen during the Interval
- that a more or less abstract Concept is mentioned, heard or alluded to during the Interval

and so on. Hence, for preciseness it may be useful to have different kinds of "relatesTo"s – a user-defined classification, or a relationship-type property for each Interval-Concept pair. Figure 5.3 illustrates this, for the chimpanzee grooming example. (The grey boxes and arrows are system-defined classes and relationships; the rest are user-defined.) However, this is a rather complicated feature for a relatively minor increase in expressiveness, so I will defer the implementation of such a scheme to further research.

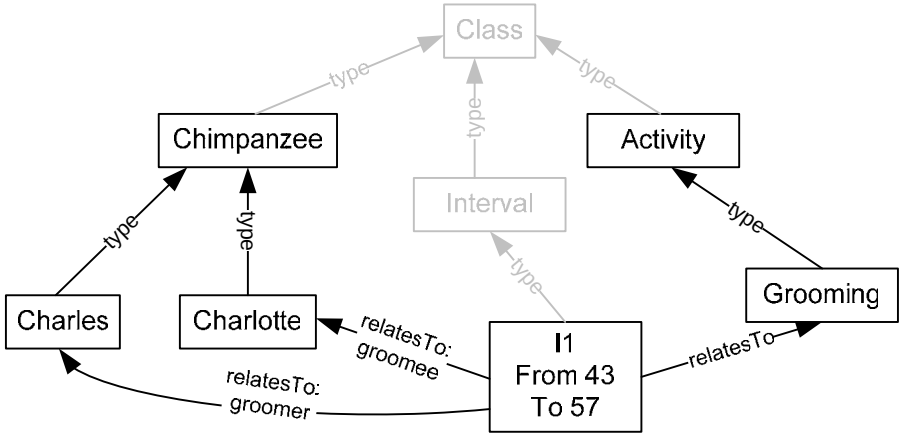


**Figure 5.3: Different kinds of "relatesTo" relationships**

### 5.1.3 Properties and model extensions

Another complementing annotation mechanism is that the MediaResources, Intervals and Concepts (and indeed any data element in OntoLog) may be described with arbitrary properties, selected or defined by the user. This is a rather common idea, used by several of the systems described in section 3.2, but the OntoLog model (heavily influenced by RDF [World Wide Web Consortium 2004a] in this regard) takes it a step further than most.

Firstly, *any* element in the OntoLog model may be described in this way, not just one or a select few, as is more common. All the elements are treated equally. Some may be more suited to this kind of description than others (it may be that Intervals are simply too numerous for the user to bother describing them much), but in the interest of flexibility, power *and* simplicity, the OntoLog model treats them all the same. A few common properties (title and description, among others)

should probably be provided as default, but this is an implementation (or user interface) issue.

Secondly, the class structure of the OntoLog model's ontologies provides a mechanism for differentiating elements, in order to describe them differently. Let us say a user creates an ontology of people and places to describe his/her video. Using the OntoLog model, this entails creating "Person" and "Place" classes (and instances of them); the OntoLog model then provides the means to say that Person individuals are described with these properties and Place individuals with those. This is not limited to ontology elements either – individual MediaResources are members (instances) of the MediaResource class, and the user may define properties for this class, and create subclasses of it. Intervals are treated correspondingly.

Thirdly, property *values* may themselves be ontology elements; they need not be string values or other kinds of literals. The model is capable of defining new objects, object types and relationship types, as discussed in section 3.1.3, through the very same ontology mechanism that is used for the stratified concept-based annotation scheme. This may be more power and complexity than most users need, but a simple default is to assume unconstrained text strings as property values, unless the user specifies otherwise.

Figure 5.4 shows an example of user-defined properties and property values. There are three user-defined properties, "favourite groomer", "age" and "main actor". The "favourite groomer" property is used to describe an individual of a user-defined class, with another such individual as its value. The "age" property is used similarly, except that its value is an integer. The "main actor" property is used to describe an individual of a system-defined class, namely MediaResource.

Note also that properties are individuals of the System-defined "Property" class. This means that they are defined using the same mechanisms as other individuals in the user-defined ontologies, and that the associations in Figure 5.1 and the arrows in Figure 5.2, Figure 5.3 and Figure 5.4 are really triplets of subject, property and property value. The "type" relationship is indeed also an individual of the Property class; this is omitted in Figure 5.4 for clarity.

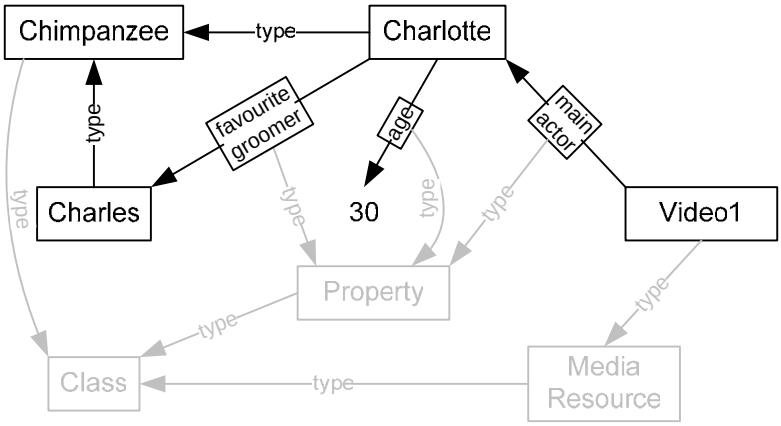


**Figure 5.4: User-defined properties**

Thus, hierarchical strata design, domain knowledge modelling and model extension (which in many ways are aspects of the same thing) are all performed using a single, uniform and reasonably simple technique. There is little difference between user-defined classes and individuals and the ones provided by default by the OntoLog model; the model extensions and domain knowledge is integrated seamlessly into the whole.

### 5.1.4 Ontologies and Projects

To facilitate reuse and sharing of ontologies – both as domain knowledge models and as model extensions – the classes of Project and Ontology are introduced into the model. This is shown in Figure 5.5.

Projects act as containers for MediaResources that are described using the same set of ontologies. For instance, for a lecture database like the one described in section 4.1.5, it would make sense to create a project for each different course. Videos from the same course would probably be described with the same ontology, while other courses might need different ontologies.

Ontologies are primarily containers for Concepts. A Project may use several different Ontologies, and an Ontology may be used by several different Projects. Ontologies also define properties that the users may describe their classes and individuals with. The properties can take text strings as values, but they can also take other individuals. Thus, yet another task for the Ontologies is to define such individuals to use as property values. This is done in exactly the same way as defining concepts; this also means that the "property values" can be used as Concepts, and vice versa. In Figure 5.4, for instance, the Concept Charles (an individual of the Chimpanzee class) is used as a property value, while in Figure 5.3 he is used as an annotation stratum, related to a temporal interval.

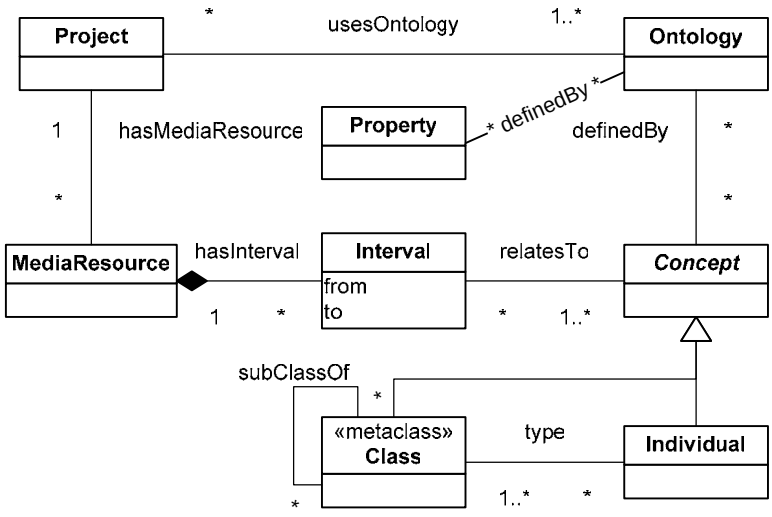


**Figure 5.5: Extended OntoLog conceptual model**

## 5.2 Scenarios revisited

In this section, the scenarios from section 4.1 are reviewed, with a discussion on how they can be implemented using the OntoLog model defined in the previous section. Two of the more demanding scenarios, the lecture database and the police investigation, are described in quite high detail; the rest are given a more cursory treatment, to avoid too much repetition.

### 5.2.1 Support for movie watching

Jon's Jackie Chan problem is easily solved with the OntoLog model. Chan is an individual belonging to several classes (Actor and Producer, at least), has several "name" or "alias" properties with different values, and is related to his films with "starring" and "producer" properties. Other classes and properties can be defined, if required.

The temporal information is also handled easily. Classes are created for each aspect of the movie in question – fighting, car chases and so on – and each class is related to a set of intervals. The intervals may form a partition, but not necessarily, and each can be described with a label or short comment, if desired. Thus, during playback it is a simple matter to select a class and jump to its next interval.

### 5.2.2 Managing interview recordings

Cathrine's requirements are also fulfilled. She wants the same topic structure for all her interviews, so she creates a project with a topic ontology. Her ontology consists mainly of a plain list of classes (topics), though a few have subclasses (subtopics). She adds her interview recordings to the project as media resources, creates intervals (segments) and assigns them to topics in her ontology. Topics and segments can be labelled and commented, and Cathrine can create a "transcription" property, if she wants to.

### 5.2.3 Support for system analysis

Ian has a more complex project, but the basic principle is the same. During the first viewing of his material, he creates a coarse ontology of people, locations and events, linking relevant intervals to them. To be more specific, he creates a Person class, a Location class and an Event class, and creates individuals of them as they are needed. Each time a significant individual person, location or event occurs in the video, Ian creates a corresponding interval, linking it to the individual.

During the subsequent viewings and analyses, the ontology is refined. Subclasses are introduced (e.g. different ranks, roles and responsibilities for persons), and individuals reassigned to them as appropriate. New individuals or base classes may emerge as the understanding of the video material improves. Properties are defined to denote the relationships between the various kinds of individuals. URLs pointing to Ian's web-based requirements database are entered as property values of relevant intervals or concepts.

### 5.2.4 Video-based system evaluation

Hallvard has two relatively independent viewpoints on his video – phases and activities – so he creates two ontologies. His phase ontology is a flat list of classes (each representing a phase), while the activities ontology is a class hierarchy three or four levels deep. He segments the video into phases by relating non-overlapping intervals to his phase classes, and also logs the various activities that occur at any time.

Having performed the annotation, Hallvard uses a simple analysis application to compute statistics on the length, number and overlap of the intervals. He also uses the annotations as a browsing index to access particularly interesting pieces of video for review. A stratification-based visualisation tool allows him to arrange the layers of intervals (a layer or strata being defined as the intervals related to a particular concept) in various orders, hide uninteresting layers and visually aggregate the layers belonging to subclasses of a given class.

This scenario has been performed in practice; the actual ontologies are described in section 7.1.1.

### 5.2.5 Lecture database

To describe his lecture videos, Steinar creates a class hierarchy of topics and activities. The top level of the ontology consists of three classes corresponding to the main topics in the course – HTML, JSP (Java Server Pages) and Databases – and three classes of "activities": Examples (when Steinar is demonstrating something), Questions (when students are asking questions) and Practical Information (when Steinar is talking about non-curricular things). He could have separated these classes into two ontologies, but he wants to keep things simple.

Steinar creates a subclass hierarchy to specify subtopics in great detail. Under JSP, he defines JSP Tags, Variables, Types, Objects, Conditions and Program Flow, to name a few. If, Loops and Method Calls are subclasses of Program Flow, and For Loops and While Loops subclasses of Loops. Steinar doesn't worry overmuch about the semantics of the ontology structure; he is happy to create Boolean Operators as a subclass of Conditions, though a Boolean operator is not a kind of condition – Steinar just reasons that everything said about Boolean Operators is also relevant to Conditions.

Likewise, Steinar doesn't care much about the distinction between individuals and classes either. His concepts don't represent actual objects or different kinds of things, just topics, and he has no plans to describe them with different properties. He wishes to add references to the course literature for each topic (at least the more specific ones), but a simple description property suffices for that.

Steinar constructs the ontology (or topic hierarchy) while logging his lectures. It is difficult to specify everything in advance, but the annotation tool allows him to add more topics as needed, and move them around in the hierarchy. The lectures are relatively quick to log, since the intervals are fairly long, and he is seldom talking about something relevant to more than four topics at the same time. He describes a few intervals with comments, but mostly he doesn't bother – there are

too many of them. Links to files containing example code is entered in a "comment" property for the media resources; it's too much work to link code snippets to individual intervals.

Searching and browsing the database, Steinar's students can review the topics they find most difficult, or see an entire lecture they missed. Analysing the annotations, they can also see what topics and subtopics Steinar spends most time on – presumably, those will be important in the final exam.

This scenario has also been performed in practice; part of the rather sizeable ontology is shown on page 158 in section 7.1.2.

### 5.2.6 Police investigation

Jostein needs to keep track of people and vehicles in his video material, so he creates an ontology with two top-level classes: Person and Vehicle. As subclasses of Vehicle, he creates classes representing different kinds of vehicles – Car, Motorcycle, Motorhome and Truck. Further subclassing is conceivable – Car could have subclasses like station wagon, sedan, coupe; or Volvo, Saab, Mercedes; or green car, red car, white car – but Jostein prefers to keep the ontology small, and instead use properties to describe the model, make and colour of the cars.

Consequently, he defines several properties and property values in the ontology as well. For the Vehicle class, he creates properties for license number, make and colour. For the Car class, he makes a car type property, and a Car Type class to use as the property's range – that is, the class the car type property can get its values from. Jostein creates a few instances of Car Type: Sedan, Coupe, Station Wagon, SUV and Convertible for starters. More can be added later, if necessary. By using the Car Type class as the property's range instead of just using a text field, Jostein avoids problems related to spelling or to using different words to represent the same concept. This is an issue especially if several different persons are describing cars: some might type "coupe" and others "coupé"; some might use "SUV" and others "sport-utility vehicle".

Annotating the video, Jostein creates individuals of Car as he sees cars in the video. He describes them as completely as possible, and connects them to temporal intervals in the videos. After some time, a sizable number of cars have been created and Jostein looks through them (possibly using some analysis tool), noting similarities. If two cars are described very similarly, Jostein reviews the video material in order to establish whether they are identical. If that is indeed the case, he merges the two cars (removes one of them and adds its properties and relationships to the other), thus establishing the car's movement in more detail.

Person types, person properties and appearances of people in the video are treated similarly.

### 5.2.7 Discussion

The OntoLog model handles all the scenarios very well, avoiding most of the problems existing systems have (described briefly in section 4.2). However, there

are some potential issues and shortcomings. One such issue concerns the semantics of the classes and individuals, the relationships between them, and how they are used in the scenarios. The most complex scenarios, Ian and Jostein, use the class structure in the "proper" way – to represent classes of things that occur in the video, and there is seldom any question about whether a given concept should be a class, subclass or individual. However, the other scenarios have a more haphazard and lenient approach to this modelling problem. Steinar is a good example; he does not really care about the distinction between classes and individuals, and uses the subclass relationship as a loosely defined subtopic link. The question is: Does this cause any problems? It would seem that the answer is no; as long as custom and resource-valued properties are not used extensively, the main difference between classes and individuals is the individuals' lack of children, and the semantic difference between subclass-of and instance-of is negligible in the scenarios we are discussing.

This, however, leads to a related problem: The atomicity or finality of the individuals. Though it is not mentioned in any of the scenarios, a mechanism for specifying a hierarchical relationship – a *part-of* relationship – between individuals may be desirable. This might be useful in Ian's scenario to specify locations at various levels of detail, e.g. that the Field Hospital is part of the Main Camp, which is part of the West Front. This could be done with custom properties, but the semantics should preferably be known by the model, so that (say) a query asking for footage of the West Front also retrieves intervals related to Main Camp and Field Hospital. This should be studied further; it might not be more complicated than including some canonical part-of property in the OntoLog model, handled analogously with the "subClassOf" and "type" properties.

## 5.3 Relevant technologies

There are many technologies that are potentially relevant for implementing the model presented above. This section presents the ones that are most important for this thesis. In subsection 5.3.1, a handful of metadata frameworks are presented; Dublin Core, MPEG-7, RDF and the ABC model, to be specific. Subsection 5.3.2 presents ontology languages and standards. (Ontologies in general are discussed in appendix A.) Subsection 5.3.3 concludes by outlining temporal databases – what they are, what they do, and how (or if) they can help with the task at hand.

### 5.3.1 Description schemes and metadata standards

There are many different standards for describing and indexing data. There is the minimalist Dublin Core, the all-encompassing MPEG-7, as well as numerous library systems, for instance the Dewey Decimal classification system and the MARC metadata format. Computer systems used in medicine often utilise ontologies developed to standardise the complex terminology, and museums catalogue artefacts according to commonly agreed-upon classification schemes. These standards are built with certain application types and domains in mind, and may not be totally appropriate for other uses, though their wide acceptance makes them desirable to use. This section describes two of the most well-known logical

models for metadata, Dublin Core and MPEG-7, as well as two metamodels, or frameworks for metadata: the ABC model and the Resource Description Framework (RDF).

### Dublin Core

The Dublin Core (DC) Metadata Element Set (DCMES) [Dublin Core Metadata Initiative 2003] is a set of fifteen descriptive elements with a specific semantic meaning, used to describe resources of any kind. The Dublin Core Metadata Initiative (DCMI [Dublin Core Metadata Initiative 2004]) is promoting and refining this concept through a series of workshops and meetings.

The fifteen elements are Title, Creator, Subject, Description, Publisher, Contributor, Date, Type, Format, Identifier, Source, Language, Relation, Coverage and Rights. All the elements are optional, and may be repeated. The data type is character string for all the elements.

Additionally, each element may be "qualified" to refine its meaning – either to indicate that its value is formatted according to some specification, or drawn from a controlled vocabulary, or to constrain the semantics of the term. For instance, the "Subject" element may be qualified to indicate that its value is a Dewey Decimal Classification code. "Relation" may be qualified as for instance "Relation.isVersionOf", "Relation.hasPart", and "Relation.isReferencedBy" to refine the semantics of the relationship. DCMI maintains a list of "approved" qualifiers – it is permitted to create others, but interoperability will of course suffer unless only the approved ones are used. See [DCMI Usage Board 2003] for the list of approved qualifiers. The qualifiers are required to be "optional" in the sense that they may be ignored without invalidating the information, though some specificity is lost. This is referred to as the "Dumb-Down Principle", and is important for the interoperability of DC tools and search engines.

DC descriptions are most often embedded in HTML pages, but may be written on index cards, put into relational databases, or encoded with any kind of scheme.

### MPEG-7

Dublin Core may be described as the least common denominator for describing (media) data. The MPEG-7 standard [International Organization for Standardization 2002] has in contrast placed itself on the other end of the spectrum: It is an all-out attempt to model absolutely every conceivable kind of metadata in a very detailed manner.

The MPEG-7 standard consists mainly of a logical model for media metadata, specified in terms of XML. The model is far too big and complex to describe fully here, but an outline of its main points follows.

The MPEG-7 model is basically hierarchical. At the bottom level, we find **descriptors**, which are “atomic” properties of something, e.g. the name of a person. Descriptors can be *primitive* (text strings, integers and such) or *composite* (e.g. histograms or lists); the “atomic” notion just means that they do not contain other descriptors; they form the leaves of the MPEG-7 description tree.

The nodes of the tree are called **description schemes**, or DSs. DSs can contain descriptors and/or other DSs, and specify relationships between them. For instance, *Person* could be a DS, containing a *Name* descriptor, as well as an *Organisation* DS through an *Affiliation* relationship. The standard specifies a horde of DSs and descriptors, and allows users to create their own DSs as well.

MPEG-7 defines its own description definition language (DDL), which is XML Schema [World Wide Web Consortium 2001a] with a few extensions. MPEG-7 descriptions are realised as XML documents, though the standard also specifies a binary encoding, if that is needed. The use of XML makes the descriptions human-readable in theory, though they tend to be so verbose and complex that this has little practical meaning.

MPEG-7 descriptions are meant to be able to cover all aspects of the media they describe. The standard defines five *viewpoints*, each with their own DSs and descriptors. These viewpoints are:

- **Creation and production** – covers for instance title, author, classification (genre, rating) and background information.
- **Usage** – access rights, owners, copyright holders, financial information.
- **Media** – the format of the described media, including encoding, bit rate, file format, amount of data and so on.
- **Structural aspects** – temporal segments (overlapping or not; possibly hierarchical) and spatial regions in audiovisual media.
- **Semantic aspects** – descriptions of "semantic objects" – events, people or concepts that are related to the described media.

The structural and semantic aspects are collectively called *content description*, while the three others are called *content management*. This division is reflected in the XML of an MPEG-7 description; these two groups (that is, their corresponding XML tags, `<ContentDescription>` and `<ContentManagement>`) are found at the top level of the MPEG-7 description tree. The tree can easily become quite deep; it is not unusual to have over ten levels, even for fairly simple descriptions. Figure 5.6 shows a fragment of an MPEG-7 description detailing the author and creation time of an image. In a complete description, the `<Creation>` tag would be on the fifth level of the tree, below `<Mpeg7>`, `<ContentDescription>`, `<MultimediaContent>` and `<Image>` tags, or even deeper.

The content description part of MPEG-7 seems to fulfil many of the requirements listed in section 4.2. It certainly has temporal and spatial expressiveness due to the many very complex DSs that are defined for structural aspects. It is also possible to create custom DSs, which makes for good semantic expressiveness, though it is unclear if the power of the DSs are equivalent to a proper ontology system. However, it suffers somewhat in flexibility – the model is more complex than is needed for many purposes, and it seems difficult to simplify it – and the usability may also be lacking.

```
<Creation>
  <Title xml:lang="en">Crushed Dreams</Title>
  <Creator>
    <Role href="">
      <Name xml:lang="en">Author</Name>
    </Role>
    <Agent xsi:type="PersonType">
      <Name>
        <GivenName xml:lang="en">Scott</GivenName>
        <FamilyName xml:lang="en">Adams</FamilyName>
      </Name>
    </Agent>
  </Creator>
  <CreationCoordinates>
    <CreationDate>
      <TimePoint>2001-10-09</TimePoint>
    </CreationDate>
  </CreationCoordinates>
</Creation>
```

**Figure 5.6: Fragment of MPEG-7 description (from [Staff and Jødahl 2001])**

### The ABC model

The ABC model [Lagoze et al. 2000], [Lagoze and Hunter 2001] is yet another approach for metadata modelling. It is presented as "an event-aware model for metadata interoperability", and has several interesting properties and goals.

A first point is its event-awareness – it realises that intellectual content evolves over time. For instance, if a book is translated, it is still in some sense the same book, though of course its language (and probably title) is changed. The author (and many other properties) remains the same, while several new properties like "translator" and "translation date" are added. This is represented in the ABC model as two resources connected by a translation event, illustrated in Figure 5.7. Events are treated like "first-class objects", and can thus have properties – the translator and translation date are connected to the translation event, not to the translated book.

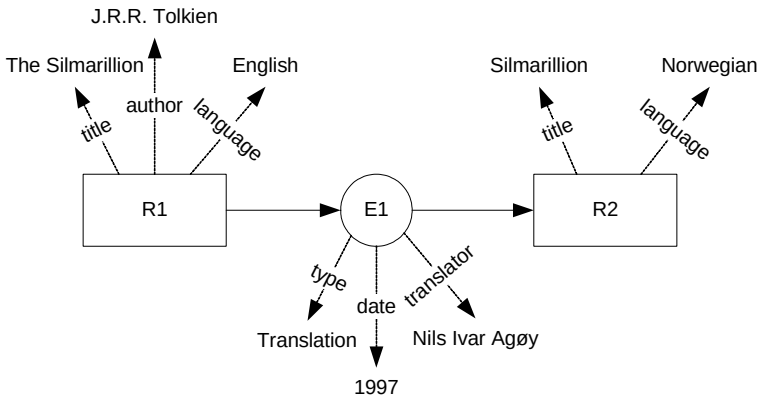


**Figure 5.7: Event-aware metadata**

This scheme enables very precise descriptions of the history of a resource, and how resources relate to each other. Each event can have several "inputs" and

"outputs", and the events and resources are in fact described slightly more detailed than in Figure 5.7; see [Lagoze and Hunter 2001] for the full details.

This is a fairly complex model (though its basic building blocks are simple). But that is some of the point: It is supposed to be able to express both simple metadata such as Dublin Core descriptions, and quite complex and detailed schemes such as the INDECS E-Commerce Metadata Model [Rust and Bide 2000] and the International Federation of Library Associations and Institutions' FRBR (Functional Requirements for Bibliographic Records) model [IFLA 1998]. In fact, the "metadata interoperability" concept mentioned earlier is exactly this: The ABC model can be used to translate between metadata standards. For instance, a community can describe books and their translations according to the pattern illustrated in Figure 5.6, and then use a process of *flattening* and *term substitution* to translate this description to (say) Dublin Core. In the case of Figure 5.6, this would entail treating the two books and the event as a single resource, figuring out what properties are relevant according to the semantics of the event and the purpose of the flattening, and mapping these properties to their Dublin Core counterparts – thus ending up with a description along the lines of this:

| | |
|---|---|
| Creator | J.R.R. Tolkien |
| Title | Silmarillion |
| Contributor | Nils Ivar Agøy |
| Date | 1997 |
| Language | Norwegian |

A third point of the ABC model is that it is meant to be a starting point or conceptual basis for communities wishing to develop their own metadata schemes. It defines a great deal of concepts that are applicable for most domains – events, resources, situations, agents, actions, roles and so on – and specifies how they should be connected. It is then up to each community to decide how they want to extend, realise and specialise the model – what kind of resources should be described, what events should be explicitly recorded and what should be ignored. For instance, a community of film historians may consider the insertion of a scene into a movie a significant event, while a casual film collector would probably not want to bother with that level of detail.

This is an interesting model, but it is perhaps most apt for heterogeneous digital libraries and museums, which need to deal with the history of their items; with versions, translations and editions; with different manifestations of the same work (e.g. "Romeo and Juliet" as play, ballet, graphic novel or film) and so on. While the model deals with temporal data, it does it in the same vein as the temporal database systems described in section 5.4.3, and not in the sense that is the topic of this thesis. It has been used for temporal annotation [Hunter 1998], but in a rather simple manner (with respect to temporal annotation) – the media material was just segmented and described with transcripts and properties. However, it was demonstrated that ABC descriptions could be mapped to both ID3 tags, HTML

pages with metatags, Dublin Core, MPEG-7 and a relational database without lossiness, ambiguity or misinterpretation.

**RDF**

The Resource Description Framework (RDF, [World Wide Web Consortium 2004a]) is a recommendation[1] by the World Wide Web Consortium designed to make the web machine-understandable through the application of metadata standards. It is a foundation for processing metadata, and consists mainly of a property-based conceptual model and XML serialisation syntax.

The basic concepts of RDF are *resources* and *properties*. A resource is something that is described by an RDF expression, and is identified by an URI. A resource is described by any number of named properties, each of which has a literal or another resource as its value. Thus, the RDF descriptions form triplets called *statements*, the parts of which are the *subject* (the resource that is being described), the *predicate* (the named property) and the *object* (the value of the property for the given subject. Another way to view the model is as a labelled, directed graph[2], with resources and literals as nodes, and properties as edges.

Figure 5.8 shows an example of an extremely simple RDF description, encoded in XML. This encodes one single RDF statement: that "http://www.idi.ntnu.no/~heggland" (a resource) has "owner" (a property) "Jon Heggland" (a literal). The `<rdf:RDF>` element signifies that this is an RDF document. The `<rdf:Description>` designates which resource is described with its "about"-attribute. The element may contain any number of sub-elements, each specifying a property-value-pair. In this case, the resource has a single property, owner, which is specified as belonging to the namespace http://www.idi.ntnu.no/~heggland/rdf/schema/jon, which (presumably) defines the semantics of the term (and whatever restrictions may be placed on its value).

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:jon="http://www.idi.ntnu.no/~heggland/rdf/schema/jon">
  <rdf:Description about="http://www.idi.ntnu.no/~heggland">
    <jon:owner>Jon Heggland</jon:owner>
  </rdf:Description>
</rdf:RDF>
```

**Figure 5.8: RDF example**

RDF Schema (RDFS, [Brickley et al. 2004]) is an accompanying candidate recommendation designed to help create RDF vocabularies. It provides a standard mechanism for declaring properties and resource types, as well as their semantics, using the RDF model. The schema description language is simply a set of resources (including classes and properties) and constraints on their relationships.

[1] A W3C "recommendation" is what most people would call a "standard".

[2] Actually a *multigraph*, since two nodes may be connected by more than one edge in the same direction.

RDF resources may be denoted as belonging to a particular class, through the use of the `rdf:type` property. Classes are resources, and may be subclasses of other classes, using the `rdfs:isSubclassOf` property. This enables a hierarchical type system similar to the type systems used in object-oriented languages such as Java. The root class of the RDFS type system is `rdfs:Resource`. `rdfs:Class` and `rdfs:Property` are two important subclasses of `rdfs:Resource`. Core properties (that is, instances of the `rdfs:Property` class) include:

- `rdf:type`, which indicates that a resource is an instance of some class. A resource may be an instance of more than one class.
- `rdf:subClassOf`, which indicates that a class is a subclass of another class (its instances form a subset of the superclass's instances). As is usual in such type systems, this is a transitive property. A class may be a subclass of more than one class.
- `rdf:subPropertyOf`, which indicates that a property is a specialisation of another. For example, the property *father* is a subproperty of the property *parent* (being a father implies being a parent). A property may be a subproperty of more than one property.

The type system is atypical in that it doesn't define classes in terms of what properties they may have, but defines properties in terms of what classes they may apply to[1]. This is done with the `rdfs:domain` and `rdfs:range` properties. For instance, an *author* property can be defined as having domain *Book* and range *Person*, meaning that books have persons as their authors (or, equivalently, that authorship is a relation between books and persons). `rdfs:range` and `rdfs:domain` are instances of `rdfs:ConstraintProperty`, a subclass of `rdfs:Property` that is used to specify constraints. The domain of `rdfs:range` and `rdfs:domain` is `rdfs:Property`, and their range is `rdfs:Class`.

### 5.3.2 Ontology standards and tools

It is assumed that the reader is familiar with the term *ontology* as used in computer science; if not, a brief introduction can be found in appendix A. There exist quite a few languages, standards and tools for defining and managing ontologies; this subsection presents a few of the most common.

#### Open Knowledge Base Connectivity

Open Knowledge Base Connectivity (OKBC) [Chaudri et al. 1998] is an API attempting to standardise access to knowledge representation systems. It is developed jointly by the Artificial Intelligence Center of SRI International and the

[1] This is a little imprecise. According to the RDF Schema recommendation [Brickley et al. 2004], `rdfs:range` and `rdfs:domain` do not specify constraints, but rather constitute a type inference mechanism: If "Stephen Donaldson" is the object of a statement, and `author` the predicate, and `author` has `rdfs:range Person`, you can infer that "Stephen Donaldson" is an instance of `Person`. However, I consider its use as a constraint mechanism more intuitive and useful for my purposes.

Knowledge Systems Laboratory of Stanford University. OKBC is designed to be an abstraction layer isolating applications from the disparities of different knowledge bases, and as such is very general, but it specifies a concrete, frame-based knowledge model:

- A *frame* represents an entity in the domain of discourse.
- A frame has *slots*, which are attributes. Attribute values can be other frames, or primitive types like strings and numbers.
- Each frame-slot-combination may have *facets*, which are constraints on the slot values. A facet may for instance assert that the value of the "parent" slot for a "cat" frame must also be a "cat" frame.
- A *class* is a set of entities. An entity in a class is an *instance* of the class. Instances that are not classes are *individuals*.
- Class frames can have *template slots* (and corresponding facets) that specify attributes and constraints for subclasses and instances of the class. For instance, the class "woman" may have a "gender" template slot with the value "female"; this ensures that all instances of "woman" have a slot "gender" with the value "female".

OKBC is complementary to the **Knowledge Interchange Format (KIF)** [Genesereth and Fikes 1992], a declarative language developed to support knowledge sharing. KIF is very expressive – more so than most actual knowledge representation systems – but does not provide operations for knowledge-base manipulation or query; hence, OKBC offers a manipulation and query API using KIF as its syntax.

### RDF Schema

RDF Schema (RDFS), described earlier, may be considered an ontology language, albeit a simple one. It provides mechanisms for defining classes and arranging them in a taxonomy, and it defines a few general classes: `rdfs:Resource`, the most general class; `rdfs:Class`, the class of resources that are classes; `rdf:Property`, the class of resources that are properties, among others. Unlike frame-based systems, slots and facets (properties and property restrictions) are first-class objects, and do not depend upon any particular class for their existence. RDFS also specifies the mechanisms for defining properties, arranging them in a hierarchy, and putting constraints on which classes they may be applied to. Thus, RDFS has expressive power enough for simple ontologies, but lacks some possibilities: for instance, to say that two classes are disjoint, or that a class is defined as the set of resources with a particular value for a particular property, or that a certain property is transitive, reflexive or symmetric.

### Ontology Inference Layer

Ontology Inference Layer (OIL) [Bechhofer et al. 2000] is a standard proposal for an ontology language based on OKBC and RDF. It tries to combine the power of frame-based systems with description logics, where a class may be defined as a

restriction on the roles it may participate in a relation as. For instance, a "carnivore" class may be defined as animals that eat other animals. If it is then specified that (members of the class) "lion" eat (members of the class) "gazelle" (a subclass of animals), it can be concluded that "lion" is a subclass of "carnivore". In contrast to OKBC/KIF, which aspires towards complete expressive power, OIL tries to establish a relatively small, common and well-defined set of constructs, to enable decidable and efficient reasoning support. OIL extends RDFS, utilising its mechanisms for class definition and constraint specification, and adding such things as cardinality, classes defined as the intersection, union or complement of other classes, classes defined as restrictions on property values (facet functionality, in other words) and transitive, symmetric and inversely-related properties. OIL has evolved into **DAML+OIL** (DAML: DARPA Agent Markup Language), which is the basis for the World Wide Web Consortium standard OWL, which is under development.

### Web Ontology Language

Web Ontology Language (OWL) [World Wide Web Consortium 2004b] comes in three flavours: OWL Lite, OWL DL and OWL Full. OWL Lite is a subset of OWL DL, which is again a subset of OWL Full.

*OWL Lite* is relatively restrictive, and is targeted at users who need "a classification hierarchy and simple constraints. […] OWL Lite provides a quick migration path for thesauri and other taxonomies". In addition to RDFS mechanisms, OWL Lite provides the ability to state that pairs of resources are equivalent, identical or different; to state that properties have characteristics such as transitive, symmetric and functional; to state restrictions on the values of a property when applied to a class; to define a class as an intersection of other classes; and some simple cardinality restrictions. Additionally, OWL Lite and OWL DL require type separation – that is, a class may not also be an individual and/or a property, and a property may take either an individual or a primitive datatype as value, but not both.

*OWL DL*[1] adds some functionality to OWL Lite: more constructs for defining classes (as enumerations, or unions or complements of other classes, and arbitrary combinations of these), and more powerful cardinality. It is more complex to implement and reason about than OWL Lite, but has been designed with a firm basis in description logics, so that it is guaranteed to be computationally complete – all entailments are guaranteed to be computed, and all computations will finish in finite time.

*OWL Full* removes the type separation restriction in DL and Lite, and thus makes it possible to treat (say) a class as an individual, providing meta-modelling functionality. This gives maximum expressivity, but loses the computational guarantee. OWL Full is a true superset of RDFS – it defines and assigns meaning to certain classes and properties, but does not introduce any restrictions or change

[1] DL stands for *Description Logics.*

the semantics on RDFS mechanisms. In contrast, only a subset of RDFS ontologies are OWL DL and OWL Lite ontologies.

**Jena**

Jena[1] is a Java API for manipulating RDF models, developed by HP Labs. It provides statement-centric methods for manipulating RDF models as sets of RDF triples as well as resource-centric methods for manipulating RDF models as sets of resources with properties. It offers import and export of RDF models in XML, N3 and N-triple formats, and supports persistent storage in SQL databases. It also supports RDQL, a SQL-like query language for RDF data, and includes a DAML+OIL layer that facilitates construction and manipulation of DAML+OIL ontologies. This only works on in-memory models, though; not on models backed by a database.

The upcoming release, Jena 2, removes this limitation, and has by and large greatly improved ontology support. It provides a unified API for the RDF-based ontology languages (RDFS, DAML+OIL and OWL) with configurable (and pluggable) inference support. Application programmers will be able to use Jena as a deductive database, asserting ground truths and axioms and letting Jena figure out the entailments in a transparent manner.

**Sesame**

Sesame[2], developed by Aidministrator Nederland, is a RDF storage facility and query engine. Like Jena, it can import and export RDF models in XML, N-triples and N3 (export only), and manipulate them through a Java API. It supports RDQL and RQL (another SQL-like RDF query language) queries, and offers a web-based RDF browsing interface. Unlike Jena, it doesn't provide any ontology or inference support (beyond the capabilities of RQL, which knows about the semantics of RDFS classes and properties), but this may supposedly be added easily as separate plug-ins due to Sesame's modular architecture.

### 5.3.3 Temporal database management systems

A common (though loose) definition of a temporal database is "a database that contains historical data" [Date 2004], in contrast to databases that contain only current data. Conventional databases[3] represent currently true propositions about the real world as tuples in relations, while temporal databases also represent propositions that have been true (or even will be true), together with the time intervals in which they are (or were, or will be) valid. One common way of looking at it (or even implementing it) is as a regular database relation variable

[1] http://jena.sourceforge.net/

[2] http://sesame.aidministrator.nl/

[3] I will limit this discussion to relational databases, as the relational model is by far the most formal, well-defined and well-understood data model.

(table) augmented with extra attributes for time information. Consider an employee database storing employee names, positions and salaries:

| Name | Position | Salary | From | To |
|---|---|---|---|---|
| Jon | Research Fellow | 23 000 | Jul 2002 | Jun 2003 |
| Jon | Research Fellow | 24 000 | Jul 2003 | Dec 2003 |
| Jon | Lecturer | 30 000 | Jan 2004 | NOW |

A conventional relation would just contain the "currently valid" tuple (i.e. the one at the bottom, in this example), while the temporal relation contains the entire employee history, with from and to dates for each tuple. Thus, it is possible to ask what an employee earned two years ago, or to list the position history for someone.

This is of course achievable within the relational model; the task of temporal database management systems is mainly to make it *convenient* to do so. Temporal relation variables need a lot of rather complicated constraints in order to guarantee their correctness, and temporal queries are correspondingly fiddly to formulate correctly. For instance, the temporal DBMS needs to make sure that the temporal information represented by the From and To attributes do not cause the relation to contain redundant or contradictory information, and a query to find the position history of employee Jon (that is, ignoring the salary) would need to "coalesce" or "pack" (the terminology is not firmly established) the two "Research Fellow" tuples into a single tuple, with a time interval of Jul 2002 – Dec 2003. However, as Date shows [Date 2004], it can be done with a relatively simple generalisation of the relational operators and a set of shorthands for temporal constraints and queries.

The **TSQL2** temporal query language [Snodgrass 1995] is essentially an extension of the SQL data definition and data manipulation languages, where tables are defined with additional features:

- Temporal tables can be "valid time" tables or "transaction time" tables, or both. Transaction time tables record the times when a row was inserted ("updates" creates new rows, and rows are never deleted), while valid time tables uses user-supplied time information.
- Valid time tables can be "event" tables, meaning that each row has an associated timestamp (or "time point"); alternatively, they can be "state" tables, meaning that each row has a time *interval* (which of course can be thought of as consisting of two timestamps), as in the example above.

"Conventional" non-temporal tables are still supported.

Temporal databases are evidently not directly applicable in the context of temporal annotations. We are not really interested in *historical* data, but rather information related to temporal intervals on a timeline that is private to each video, and has no clear relationship to "real-world" time. However, temporal

DBMSs may still do some good: TSQL2, for instance, defines a `PERIOD` data type[1] that represents a temporal interval with start and end times, and provides operators for working with it. There is no reason why non-temporal DBMSs could not provide such a data type, however.

In any case, temporal DBMSs have not matured beyond the prototype stage. The few implementations that exist (e.g. Tiger [Bukauskas 2003]) are usually implemented as a layer on top of a SQL DBMS.

## 5.4 Technology choices

This section presents my technology choices for the OntoLog model. While this may seem an implementation issue better suited to a later chapter, deciding which (if any) of the metamodels presented in section 5.3.1 is actually more of a conceptual design choice than a physical implementation decision. Hence, it is presented here. First, I discuss the choice of RDF and RDFS as the basic technologies to build an OntoLog system on, and the consequences of this choice. This is followed by a discussion on MPEG-7 and temporal databases, two technologies found inappropriate for this endeavour.

### 5.4.1 RDF

The Resource Description Framework (RDF) [World Wide Web Consortium 2004a] is a good candidate for the implementation of an OntoLog system. Indeed, the OntoLog model was inspired by RDF, though the techniques of describing things with user-defined properties, creating ontologies of resources and organising all this in a semantic network [Tsichritzis and Lochovsky 1982] are not ideas exclusive to RDF.

RDF provides a lot of advantages for implementing an OntoLog system:

- Its fundamental model makes it very extendable. Not only is it simple to add new properties, classes and objects; it is easy to build upon existing data structures and schemas in a backwards-compatible way, without needing to edit that which is built upon.
- It is simple enough that users without backgrounds in computer science may be able to understand it. Though complexity can be hidden by well-designed user interfaces, it is advantageous that the user has a clear model of what goes on "behind the scenes".
- It provides default mechanisms for handling type systems and ontologies. Additionally, the structures used for these purposes are the same as what is used for any other kind of information stored in RDF. This makes the handling of RDF models very uniform and orthogonal.

[1] Note that the `INTERVAL` datatype provided by SQL does not in fact represent an interval in the normal sense of the word, but rather a *duration*, e.g. 3 days.

- Since it is a popular standard, ontologies and ontology tools are publicly available on the web for use and reuse.

Several of the elements of the OntoLog model in Figure 5.1 have direct counterparts in RDF, namely class, type and individual. The other classes and associations are readily implemented as RDF classes and properties. The `from` and `to` attributes will also have to be properties, with literal values. RDF (or to be precise, RDFS) provides standard functionality for labelling and describing resources (the properties `label` and `comment`). In addition to these, I intend to provide the Dublin Core properties as default properties for the users to describe their resources with, if they do not wish to create their own.

RDF resources are identified by URI; this provides a simple mechanism for linking the actual media data to the MediaResource instances: The URI of each MediaResource resource is an URL pointing to the media data.

RDF is not without its drawbacks. Due to the way absolutely everything is stored as subject-predicate-object triplets, even simple queries will produce a lot of joins if the data is stored in a relational database in the "natural way" used by e.g. Jena. A query to find all the video intervals a certain concept is related to, will involve joining the statement relation to itself four times. This may be the price to pay for flexibility.

A more efficient implementation could perhaps be achieved through the use of an object-oriented database system (OODBMS). The core classes like Interval and MediaResource could be modelled with explicit attributes, instead of relying on the more general but less efficient RDF statement mechanism. However, this would sacrifice some flexibility, usability and interoperability. For technical reasons, it is in general harder to modify the schemas in OODBMSs on the fly than it is in RDF. OODB schema design is also a task normally given to DB experts, and the typical tools and user interfaces designed for it reflect this.

### 5.4.2 RDF Schema

RDF Schema (RDFS, [Brickley et al. 2004]) is chosen as the ontology language for my implementation of an OntoLog system. An RDF-based language is a natural choice; however, the semantics provided by RDFS are rather crude and simple compared to other ontology languages. Luckily, most of these others are extensions of RDFS. DAML, OIL, DAML+OIL [Bechhofer et al. 2000] and the emerging OWL standard [World Wide Web Consortium 2004b] are all either based on RDFS, or easily translated to RDFS, and all are easily represented in RDF.

RDFS provides the most fundamental elements of ontology languages – the notions of class, subclass and instance – and this is sufficient for many annotation purposes. Languages built on RDFS are easily handled in the OntoLog model, just like any other model extension. Tools and user interfaces take advantage of being aware of significant semantics, though. For instance, alternative methods for defining classes as (say) complements or intersections of other classes may be very useful in some applications. However, I will stay with RDFS for simplicity

reasons (and for tool support during coding) and leave the study of OntoLog models with more powerful ontology languages to future research.

### 5.4.3 MPEG-7

MPEG-7 may seem to have covered everything concerning the description of multimedia data, so creating yet another annotation model could be considered a reinvention of the wheel. However, MPEG-7 has several properties that in my opinion make it less than perfectly suitable for the purpose at hand.

First of all, it is a very big standard, and only a relatively small part of it deals with semantic annotation, which is the focus of this thesis. Most of the standard is concerned with low-level features of the media, with rights and ownership, and with bibliographic data focused towards the TV broadcasting domain.

Secondly, it is a very complex standard. Weeks (or perhaps months) of study are needed to get a proper overview of it, and even simple descriptions need a great deal of syntax and structure. Tools and user interfaces can hide this complexity from the end users, but only to a certain degree.

Thirdly, the XML notation specified by the standard is cumbersome to manage. It is very verbose, and arranges the data in a hierarchy that is in my opinion unsuitable for the kind of temporal annotation I wish to support. For instance, to describe an interval semantically, the interval must be defined in the "structural" branch of the MPEG-7 tree, and the semantic descriptions in the "semantic" branch – conceptually close, but syntactically extremely disjoint! In fact, it is very likely that an MPEG-7 application would benefit from managing the data in internal structures that correspond better to the purpose of the application, and just use the MPEG-7 XML format for interchange. After all, XML is for data interchange, not data management – hierarchical databases were rightfully abandoned decades ago.

But in that case, I find it not really worthwhile to consider MPEG-7 much within the context of this thesis. When (or if) MPEG-7 gains popularity and tool support, it should be a relatively simple matter to create an adapter that transforms the OntoLog model into MPEG-7, thus gaining the benefits of standards compatibility.

### 5.4.4 Temporal DBMSs

A temporal database management system (DBMS) is one with explicit support for time (beyond simple `DATE` and `TIME` datatypes). Since we are dealing with temporal data here, it might make sense to consider this while designing and implementing the model. However (as is discussed in section 5.3.3), what is usually understood by the term "temporal data" is *historical* data – e.g. in what department did employee X work in February 2002? – and that does not match very well with what video annotation systems want to do.

It might be possible to force a fit: One could possibly say that a tuple consisting of a MediaResource and a Concept has a "valid interval" corresponding to the Interval attributes in the OntoLog model. However, that seems a little construed

and awkward. For one thing, "valid intervals" in temporal DBMSs are implicitly assumed to refer to "real time" – the single, universal timeline ranging from the Big Bang to the end of the universe – while in the temporal annotation domain, we would need the intervals to refer to *playback* time, which is separate and different for each video, and has no simple mapping to real time. We have little use for dates, for instance; few videos are longer than a day, and even if they were, it would make no sense to connect a Concept to (say) the interval 2004-04-16T16:33:00–2004-04-16T16:35:00 – the intervals must be relative to the beginning of the video, which of course corresponds to no single instant in real time. TSQL2 [Snodgrass 1995] provides a mechanism for defining custom *calendars*, but it is not clear if this facility can be used to implement proper playback time intervals, or whether it just provides an alternative view of real time.

Temporal DBMSs typically also provide a proper interval data type[1] (in TSQL2 called `PERIOD`), with operators for temporal algebra and comparison. But again, it seems that such intervals (at least in TSQL2) are assumed to refer to real time, and include a date – thus making them awkward to use for our purposes. For these reasons, I choose not to consider temporal DBMSs further as an implementation strategy for an OntoLog system.

## 5.5 Summary

In this chapter, I have presented the OntoLog model for semantic video content:

- Its requirements and design goals – flexibility, extensibility and simplicity.
- How it can be used in practice – examples of its application in realistic scenarios.
- Its construction and properties – based around RDF, with RDFS ontologies as annotation strata.

However, that is but a part of the design of a usable system for video annotation; equally important is the design of tools and user interfaces for manipulation of the model. That is the topic for the next chapter.

[1] Recall that SQL's `INTERVAL` might more properly be called `DURATION`.

# 6 Utilising the OntoLog model

As mentioned in chapter 2, an annotation model in isolation is little more than an academic exercise. It is hard to fully envision (not to mention evaluate) the power and capabilities of the OntoLog model without some tools and user interfaces to present a concrete implementation of it. More importantly, there may be great merit in exploring the possibilities for novel and interesting ways of interacting with video annotations, enabled by the use of ontology-based stratification and RDF. Hence, this chapter investigates how the OntoLog model may be exploited fully; how the OntoLog model's novel concepts impact the design of tools and user interfaces. The chapter is organised according to the various tools that have been designed and implemented, culminating in the novel information gathering application Savanta.

A general discussion on the impact of ontologies on tools for temporal annotation is given in section 6.1. Section 6.2 presents *OntoLog*, the application the whole system is named after, which provides functionality for creating the metadata in the first place, as well as intra-video browsing and presentation, and simple analysis. This is followed by a discussion about automatic and semi-automatic generation of annotations, and how this can be integrated into the OntoLog system and the OntoLog application, in section 6.3. Section 6.4 presents a possible extension to the OntoLog model and application, enabling spatial annotations.

Section 6.5 presents *OntoLog Crawler*, a web-based searching and browsing tool for both OntoLog data and other kinds of RDF data. *Ana*, a tool for more advanced temporal analysis of OntoLog annotations is described in section 6.6. These two tools are presented rather briefly, because they are early explorations of ideas more thoroughly treated in the most interesting application in this chapter: Savanta.

Section 6.7 presents *Savanta*, an application embodying a novel paradigm for information gathering in temporal annotation databases. It integrates search, analysis, visualisation and navigation seamlessly, building on the browsing ideas of the OntoLog application and OntoLog Crawler, the analysis and algebra capabilities of Ana, and modern interactivity paradigms. The two subsequent sections present alternative search tools, *Savantoogle* in 6.8 and *Forms* in 6.9, created in order to perform a comparative evaluation of Savanta. Section 6.10 summarises the chapter.

A note on naming: I use the term "the OntoLog system" to refer to the entire system I describe in this thesis, consisting of "the OntoLog model" – the conceptual model underlying the system – and the various tools built around it: "the OntoLog application" (sometimes called just OntoLog for brevity, though this term is most often used as a synonym for "the OntoLog system"), Savanta, OntoLog Crawler and others described in this chapter. Finding good names is difficult; it has been said that data modelling is mainly a matter of finding agreeable names for things [Carlis and Maguire 2001]. I hope the prolific use of

the OntoLog name for several different (though related) things does not cause confusion.

## 6.1 The impact of ontologies

The decision to use ontologies has clear implications for the design of the tools supporting the OntoLog model. For one thing, an interface for constructing and editing ontologies is needed. However, I choose to focus more on another point: the possibilities and complexity introduced by the explicit semantic relations between the concepts of an ontology. Take for instance the lecture scenario from sections 4.1.5 and 5.2.5: Say that a particular interval A is related to the programming language concept "for loops". In the ontology, "for loops" is a subclass of "loops", which in its turn is a subclass of "flow control" (and so on). If this is to be of any use, the tools and user interfaces must be able to reason about this: to infer that when the interval is related to "for loops", it is also *implicitly* related to "loops", and thus also to "flow control" and so on, up though the class hierarchy. The same goes for when the interval is related to an individual: it is implicitly also related to the class(es) the individual belongs to.

This reasoning or inference mechanism, simple as it may seem, is the main reason for using ontologies. It makes it possible to annotate video with very detailed concepts, without having to worry about being too specific: you get the more coarse-grained annotations "for free" once you have organised the concepts in generalisation-specialisation hierarchies. Users performing a search after "flow control" will get interval A as a result, even though the annotator did not explicitly connect the interval directly to the "flow control" concept. In other words, the annotations can be created at one level of detail, and used at other (less specific) levels of detail.

Another reason for using ontologies is that it enables the users to describe the objects and concepts relevant to their video material with arbitrary level of detail, from simple topic lists to complex networks of classes and individuals, described according to their type.

Of course, the tools and user interfaces must be designed to take advantage of this. Search and query tools must be programmed to take advantage of the ontology structure. Presentation tools (for editing, browsing and/or analysis) must provide the user with controls for moving between different levels of detail, and for navigating through the ontology. Displaying the structure of an ontology in conjunction with using its concepts as annotation strata is both a challenge and an opportunity.

In the remainder of this chapter, several tools and user interfaces are presented, elaborating the impact of ontologies and RDF on the production and use of semantic temporal content annotations.

## 6.2 OntoLog – annotation editor

OntoLog is the "main application" of the OntoLog system, in that it is the producer and editor of the annotations and ontologies. It provides an ontology

editor, an interface for describing media resources with non-temporal metadata, and an interface for manual annotation/logging and simple analysis of video content. An early version of OntoLog is also described in [Heggland 2002], which is included as appendix C.

### 6.2.1 Objectives and requirements

The fundamental objective of the OntoLog application is to enable the creation of ontologies, descriptions and annotations. Hence, it needs an ontology editor, as well as interfaces for creating and organising projects and describing media resources.

That is rather trivial; what is more interesting is the need for a logging interface, a mechanism for defining intervals on the media resources, and relating them to concepts in the ontologies. As discussed in chapter 3.3, quite a few approaches to this kind of task have been tried, but not with ontology-based stratification, which may provide some novel possibilities.

An interface for creating temporal annotations must of course also provide the means for editing them. This requires a mechanism for viewing and selecting them; a "configurable visualisation of the intervals and the ontology" as specified by the Hallvard scenario in sections 4.1.4 and 5.2.4 is my starting point for designing this. This visualisation should take advantage of the ontology semantics, to alleviate the topical granularity problem discussed in section 4.2.3. In other words, it should provide a configurable level of detail, as suggested in section 5.1.2, which also will facilitate better overviews for skimming, browsing and navigation.

Other more trivial objectives include a tight integration between the logging interface and the ontology editor, so that the ontology can be constructed and changed during logging, as required by most of the scenarios. It is also an obvious requirement that the logger interface should be convenient for navigating the media material, based on the annotations or just the timeline.

### 6.2.2 Design and implementation

OntoLog's logging interface is shown in Figure 6.1. The left panel contains the ontologies the user is working with. The classes and individuals are organised in directed acyclic graphs (DAGs), since the RDF Schema standard supports multiple inheritance and typing[1]. However, the ontologies are presented as trees, for compactness and usability reasons. The right panel displays a horizontal timeline with the annotation strata corresponding to each concept in the ontology, as described in section 5.1.2. The timeline – two minutes and thirteen seconds long in this example – goes from left to right in this panel, with a vertical red line

[1] Cycles are allowed in the RDF Schema recommendation – the semantics are that if two classes are subclasses of each other, they are the same and equivalent – but OntoLog does not support this, for simplicity.

indicating the current media position. (The actual video is shown in a separate window.)

Each stratum consists of a series of interval lines along the time axis, indicating where in the media resource the concept is relevant or present. The strata corresponding to collapsed concepts (concepts with subconcepts that are not currently displayed in the tree) are shown as of lines of varying thickness. This is because they represent an aggregation of the strata beneath them in the hierarchy. This is somewhat akin to the merging and disaggregating functionality of OVID [Oomoto and Tanaka 1993], but OntoLog does it with concepts instead of scenes, and utilises the defined semantics of concept relationships. Thicker line segments indicate that several subconcepts have intervals that overlap temporally. For instance, the Female class in the "Chorus parts" ontology shown to the left in Figure 6.1 have two subclasses, Soprano and Alto (visible in Figure 6.2). The media in this case is a concert recording of Antonio Vivaldi's "Domine Fili Unigenite"[1] and we can see from the annotations that the two female voices are singing for about two thirds of this song – approximately one third together, and one third separately. The line thickness is directly proportional to the number of concepts it represents. Figure 6.1 only shows interval lines of thickness 1 and 2; a greater variation can be found in the screenshots in the section about Savanta (section 6.7).

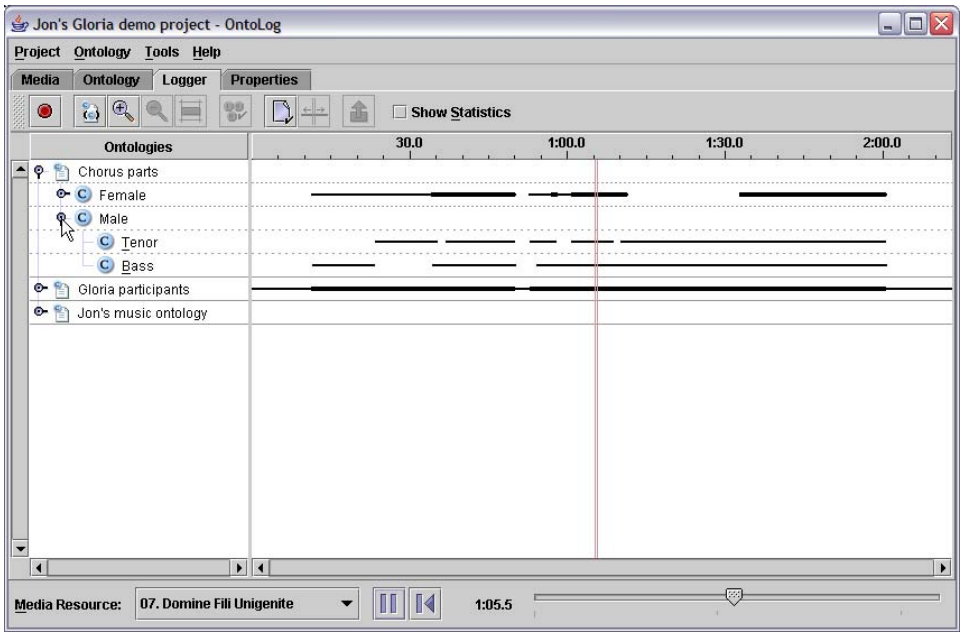


**Figure 6.1: OntoLog's logging interface**

This visualisation also has a superficial resemblance to LifeLines [Plaisant et al. 1996], which also uses lines of varying thickness, but LifeLines uses the thickness to indicate a time-varying aspect of some interval (e.g. amount of medication

[1] From his work "Gloria" for choir and orchestra.

administered to a patient in a medical record), while OntoLog uses it for aggregation; OntoLog also has a more general aggregation functionality that is usable for quite deep hierarchies, while LifeLines corresponding summarisation technique seems to need more customisation and effort to be effective.

The Logger panel has two modes, *browsing* and *logging*. In logging mode, each concept in the ontologies can be clicked on and off (using the mouse, or with keyboard shortcuts) during playback or media navigation, thus creating intervals linked to the concepts. Several concepts can be "active" at once – that is, there is no limit on the number of intervals that may be under creation at any given time. Thus, a skilled logger can log overlapping occurrences of concepts or different aspects of the video material simultaneously.

The ontology can be reorganised, edited and extended during logging. OntoLog provides VCR-like controls for directing media playback, and annotation intervals can be edited by direct manipulation, and described with properties.

The logger panel also provides a SMIL export function. This produces a SMIL file [Rutledge 2001], [World Wide Web Consortium 2001b] specifying a "virtual edit" of the selected media resource, namely a concatenation of the intervals related to the currently selected concept. Thus, you could very easily create a SMIL version of the "Domine Fili Unigenite" song with just the parts where the female voices are singing.

The ontology editor is shown in Figure 6.2. It provides mechanisms for definition and description of concepts, and import/export of RDF files in XML format. The interfaces for describing media resources, projects and properties look quite similar; the interested reader is referred to the OntoLog home page (contact the author).

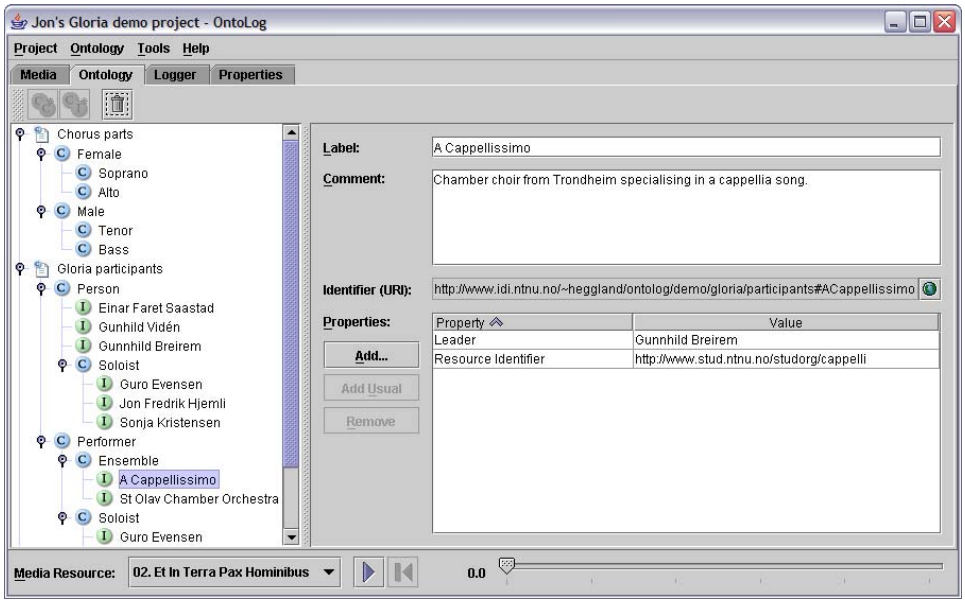


**Figure 6.2: OntoLog's ontology editor**

### 6.2.3 Discussion

The logging interface has been iteratively developed in close cooperation with actual users. It is well-liked and efficient; informal studies based on the Hallvard scenario (sections 4.1.4 and 5.2.4) have shown that with a reasonably small number of relevant concepts, only one or a few passes through the clip is necessary for an adequate set of annotations.

It also constitutes a powerful browsing interface. The hierarchical aggregation of the strata intervals provides a visual index and summary of the annotations – with most or all of the concepts collapsed, the display shows how "thickly" the video is annotated, and which concept subtrees are most important. By expanding and collapsing subtrees, users can concentrate on the concepts that are most relevant to the task at hand, and hide non-relevant information.

The main drawback of the logging interface as a browsing tool is that it only shows one video at a time. It is very useful for intra-video browsing, but not for inter-video browsing. It also offers no search, query or filtering functions. However, these shortcomings are addressed later in this chapter.

## 6.3 Improving annotation production

The OntoLog application provides a user interface for creating annotations manually. This is time-consuming work, and systems relying on it may be criticised as being only of academic interest because of this – who can afford to spend so much time describing video? There are several answers to this critique:

- Automatic annotation based on techniques such as digital image processing and speech recognition [Chua et al. 2002] , [Oard 1997], [Le Saux et al. 2003] works to a certain degree, but are often a bit limited, domain dependent, and have a hard time capturing high-level semantics.
- Research focusing on conceptual modelling, presentation and retrieval does not need to care too much about how the data are produced to begin with – that is just a single (and fairly independent) part of the system as a whole, and a completely different research area to boot.
- Some people are in fact willing to spend a lot of time annotating and indexing video; many are doing it by hand on paper, so any kind of computer assistance is probably helpful.
- It is a moot point: User interfaces for manual annotations are in any case useful for augmenting, reviewing and polishing automatically created annotations.

However, it is obvious that techniques for automatic or semi-automatic annotation, or for easing and speeding up the manual annotation process, would be very useful for OntoLog.

### 6.3.1 Visualisation for prediction, browsing and navigation

A study of possible techniques for improving annotation production in OntoLog is described in [Stengel 2003]. This project resulted among other things in a prototype implementation of waveform visualisation and video thumbnails in the logger interface, as shown in Figure 6.3, as well as other user interface improvements. This helps a logger anticipate changes in the video material, and thus log more efficiently and accurately. It also provides structure-based overview and browsing facilities.

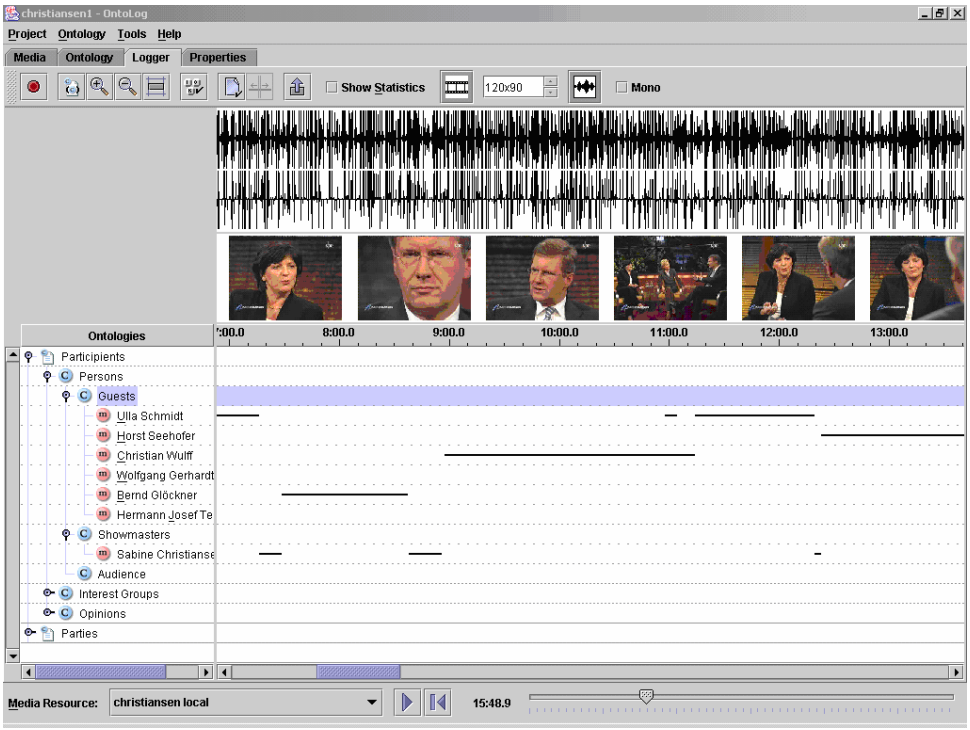


**Figure 6.3: OntoLog Logger panel with waveform and thumbnail display**

A possible improvement of the thumbnail scheme would be to include a "graphical scrollbar" or video slice bitmap, like the one presented in [Liou et al. 1999a]. This technique, illustrated in Figure 6.4, consists of copying a row and/or column of pixels from each frame of video, and concatenating them to create a long bitmap representing the video as a pattern of colours. This kind of visualisation has several nice properties: it is compact, has a high temporal resolution (each frame is represented), it shows colour distribution and aspects of the frame content (if you know how to look), and shot boundaries are easily identifiable as discontinuities in the bitmap. This would alleviate the resolution problem of the thumbnails somewhat, and would make it easier to anticipate scene changes.

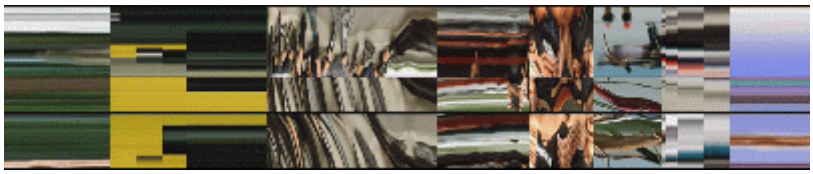

**Figure 6.4: Video slice bitmap (adapted from [Liou et al. 1999a])**

### 6.3.2 Taking advantage of automatic segmentation

Algorithms for automatic segmentation of video into shots and audio into phrases are relatively common and well understood ([Zhang et al. 1993], [Arman et al. 1994b], [Boreczky and Rowe 1996], [Arons 1997], [Kimber et al. 1995]).

This can be taken advantage of in OntoLog. Granted, such techniques as referenced above are based on the structure or syntax of the media material rather than on its semantics, which is OntoLog's focus. However, the physical structure of video is often directly related to its semantics: scene changes often signify changes of topic, and shots may correspond to changes of speakers in a discussion. In any case, some user interaction is needed, since it is not likely that a straight segmentation of the media material is sufficient annotation.

One problem with such segmentation is that the video and audio segmentation do not necessarily match. For instance, a news anchor may keep talking uninterrupted while the scene changes from studio to location, or he/she may pause and change the topic without any corresponding shot boundary. This complicates the segmentation – should we put segment boundaries only at instants where the video and audio "are in agreement"? Or at any time when there is a boundary in either the audio or the video? Fortunately, the OntoLog model permits such different viewpoints to coexist and enrich each other.

Figure 6.5 shows how automatic segmentation could be integrated into OntoLog. (This was not actually implemented, due to time constraints, and this thesis's focus on modelling and user interface issues as opposed to automatic video/audio classification.) The user could select "Perform Segmentation" from the Tools menu, whereupon OntoLog would add the "Automatic segmentation" ontology to the project, and perform the segmentation by adding intervals to the leaf concepts in the ontology. Since OntoLog is interval-oriented, it could use alternating intervals of "Video segment A" and "Video segment B" to represent the shots in the video; audio segments are treated similarly.

With this segmentation as a starting point, the video could be annotated by dragging and copying intervals from the segmentation ontology to a semantic ontology. It is likely that many of the intervals related the semantic ontology will have endpoints that correspond to those of the segmentation intervals – audio segments will probably be strongly correlated to speakers in a talk show, for instance. The segmentation might be too fine-grained, but OntoLog provides quick and simple mechanisms for joining intervals. In this way, the manual annotation process is reduced from watching the video continuously, turning concepts on and off, to skipping from segment to segment, determining what

semantic concept(s) it belongs to, and moving it there. This should greatly diminish the amount of time and effort needed, as well as make the annotations more accurate.

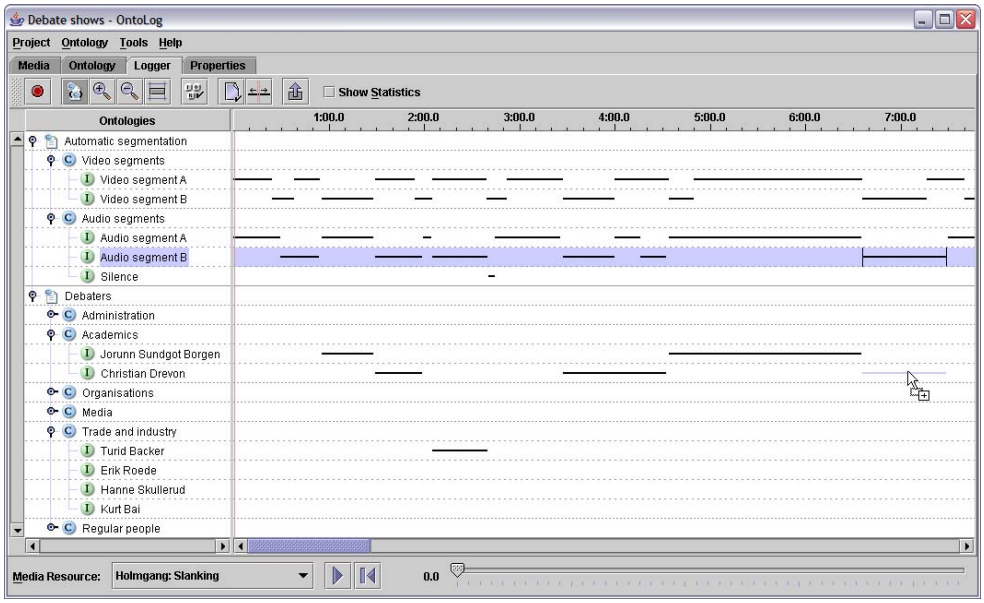


**Figure 6.5: Automatic segmentation in OntoLog**

### 6.3.3 Advanced analysis for classification and recognition

A natural extension of the functionality described in the previous section would be to have the system assign the segments to semantic concepts. This is significantly harder, since it requires the computer to understand, at some level, what it is "seeing" or "hearing". However, reasonably successful systems for video and audio classification have been made [Kazman et al. 1996], [Slaughter et al. 1998], [Foote et al. 1998], [Lu 2001], [Chua et al. 2002], [Le Saux et al. 2003]. It is for instance possible for a computer, if the signal quality is good, to determine whether an audio signal is music or speech, whether the speaker is a male, female or child, and (with sufficient training) to determine the identity of the speaker. Likewise, video classification algorithms can distinguish between outdoor and indoor scenes, between cityscapes and nature; they can recognise talking-head news, or different kinds of sports based on movement patterns, colours and line patterns on the ground; and they can recognise the faces of individual people.

This could be utilised by OntoLog: Depending on the algorithm, an ontology like the "Automatic classification" ontology in Figure 6.6 could be introduced in OntoLog, and intervals created and assigned to its concepts automatically. The hierarchical organisation and semantics of OntoLog's ontologies is eminently suited for this: Classification algorithms may have varying degrees of success and confidence, for if (for instance) a voice recogniser fails to identify a particular voice segment, it may at least be able to determine if it is a female or male voice,

and relate the corresponding interval to (say) the “Male speech” class instead of an individual of that class. Thus, a simple and natural fallback mechanism is provided.

The classification produced by such a system might not exactly match the needs of the annotator, but it would be very simple and fast to move the intervals to the desired ontology, or perhaps just edit and extend the generated classification ontology.

For specialised domains, e.g. news video, even more specific automatic annotation is conceivable. Text appearing on screen – headlines and names – could be recognised and interpreted, making it possible to create concepts for individual news items and people. It might be possible to recognise jingles and logos, and separate a news broadcast into domestic, foreign and economic news, commercial breaks, sports and weather forecasts [Chua et al. 2002]. However, such algorithms would have to be tailored and tuned to the desired domain, and would probably not be very useful outside their speciality.

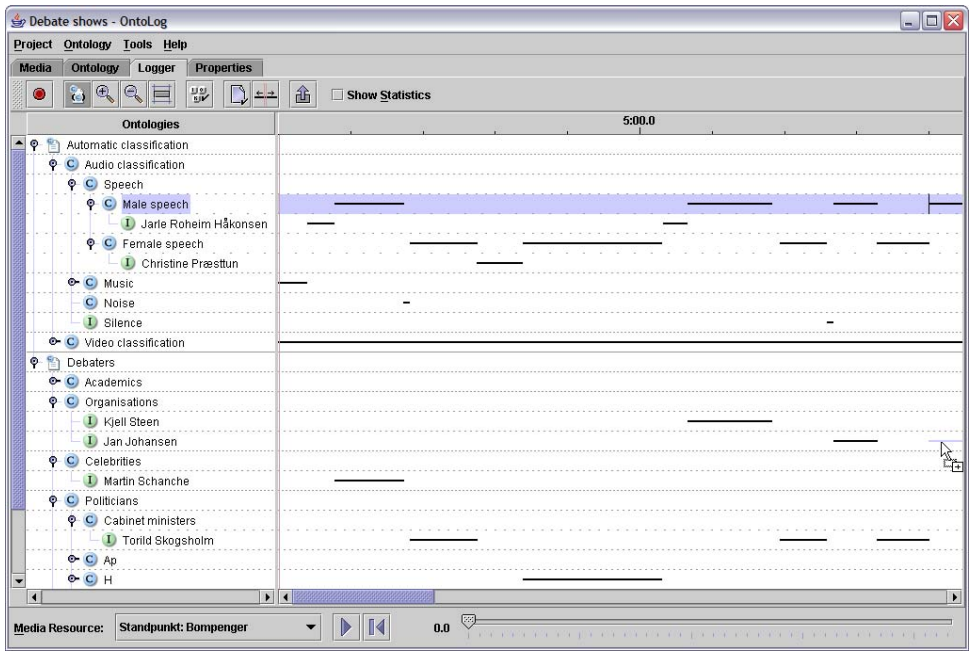


**Figure 6.6: Automatic classification in OntoLog**

### 6.3.4 Summary

Manual annotation with OntoLog (as with other tools, for that matter) is time-consuming, but can be made easier and faster by relatively simple means. Well-tried techniques for video and audio segmentation and classification can be integrated into OntoLog’s Logger in a non-intrusive and natural way, without changing the model or affecting other parts of the system. The ontology-based annotation model provides a flexible, adaptable and forgiving framework for integrating the results of multiple analysis techniques.

## 6.4 Spatial extension

Spatial annotations in video databases are not very widespread. As mentioned earlier, they are not as necessary as temporal annotations, since spatial/visual information is much easier and faster to assimilate than temporal information. Nevertheless, we have performed a study into spatial extensions for the OntoLog model as a Master thesis. A full account of the project is given in [Litsheim 2003]; it is summarised in this section. A comprehensive treatment of spatial annotation is beyond the scope of this thesis, but the following discussion gives a rough introduction, and a suggestion as to how a fairly simple yet expressive spatial expressiveness may be incorporated into the OntoLog model.

### 6.4.1 Objectives and requirements

A study into the state of the art in spatial annotation showed that this is a relatively immature research subject. As indicated in chapter 3, very few video database systems address the subject at all. In addition to BilVideo ([Dönderler et al. 2003], section 3.2.10), VideoQ [Chang et al. 1997] is such a system; however, it focuses on visual, low-level features – colours, textures, shapes, motion – and uses just a simple keyword scheme for high-level annotations. RAVEN [Schoepflin et al. 2001] and VideoTalk [Liou et al. 1999b] are little more than test rigs for automatic object tracking techniques, and have no query functionality. VideoAnnEx [IBM Research 2002], IBM's MPEG-7-based video annotator, has spatial capabilities, but uses a fixed, shot-based segmented annotation scheme and static regions. Its conceptual model is very plain and simple, and it offers neither querying nor playback capabilities geared towards spatial annotations.

The state of the art indicates that spatial annotations are useful for two purposes: For query/retrieval, and for "enhanced playback" – visual indication of salient objects during the actual playback of the video. All the systems mentioned above support one or the other (more or less), but none support both. Another finding is that spatial annotation is a time-consuming and tedious process, even when (semi-) automatic techniques are employed. The segmentation and tracking algorithms of RAVEN need about two seconds per video frame. Additionally, a significant amount of user input is needed to adjust the bounding shapes and help the system in ambiguous cases, and this requires some familiarity with how segmentation and tracking algorithms work.

We established the following loose requirements for a spatial extension to the OntoLog model:

- The model should support dynamic bounding shapes connected to concepts in an ontology, analogous to the non-spatial annotations in OntoLog. In this way, a definite connection between objects seen in the video and high-level information is established.
- The model should be geared towards manual annotation, since this thesis is concerned with modelling and user interfaces, rather than with image processing algorithms. User input should not be required for every frame

of video; if motion characteristics are simple (for instance linear), it should be possible to record it with a minimum of effort.

- The model should enable visualisation of object bounding shapes on the video during playback. This may seem obvious, but the BilVideo model, for instance, does not support it, as it only records the centres of the bounding rectangles, not their extents.
- The model should be able to record enough information that useful queries on object trajectories, distances, speed and spatial relationships can be performed.
- The model should be easy to integrate with the rest of the OntoLog system. This entails the use of RDF. Spatial indexing schemes like R-trees might be useful for the queries mentioned in the previous paragraph, but we chose to defer questions of query efficiency to a later time, and focus rather on the "enhanced playback" function of spatial annotations.

### 6.4.2 Design and implementation

The simplest method for recording the size and position of an object in a video interval is to store its bounding shape for each frame of the video. However, this creates a lot of unnecessary information when the object is static, in which case a single shape with duration would be sufficient. If the object moves, things become more complicated, but not necessarily much: If the object moves with constant speed in a straight line, a shape with duration and a motion vector will be sufficient. To determine where the object is at a given time, the system just has to translate the shape along the motion vector. If the object moves non-linearly, or changes its movement characteristics, its motion can be approximated by segmenting the motion into intervals where the motion is fairly linear. A similar technique can be used for changes in the size and shape of the object, at least for simple bounding shapes like ovals and rectangles.

Based on this observation, we decided to build the model on this concept of linear interpolation. However, instead of using motion vectors, we decided to use *keyshapes* – shapes indicating the shape and position of an object at the start and end of a segment of linear movement/transformation. This contains enough information to perform an interpolation for the intervening frames, and is easier to understand and manipulate for the end user than a vector-based model. Figure 6.7 illustrates this scheme. At 100 ms, the position and shape of Duke (Java's mascot) is represented by a bounding rectangle with coordinates and size. At 400 ms, Duke has moved and become slightly smaller, and the other keyshape rectangle represents this. At the intervening times, Duke's position and size are approximated by rectangles computed by linear interpolation of the two recorded rectangles. This enables the user to represent simple movement in a simple way. If Duke started moving in another direction, a third keyshape could be added; for best results, keyshapes should be added when movement characteristics change. This scheme doesn't work as well when objects move in curves, but their motion can be approximated as finely as needed – in the most extreme case, a keyshape for every frame could be used.

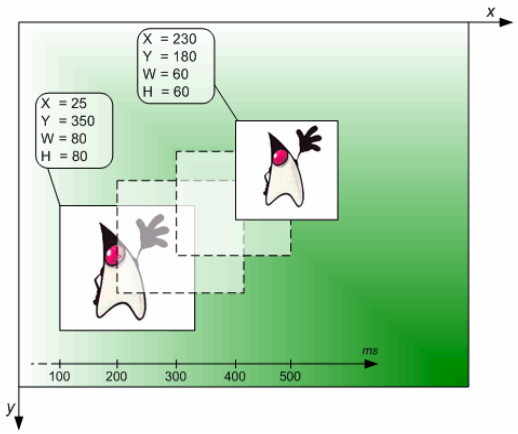


**Figure 6.7: Keyshapes and interpolation of linear movement and transformation**

Figure 6.8 shows the OntoLog model extended with keyshapes for spatial annotation. Two new entities have been introduced: *Keyshapes* and *Keyshape*. A Keyshape is a timestamped shape that is specified by four numbers: x, y, width and height. This is sufficient for rectangles and ovals; we chose to ignore more complex shapes like arbitrary polygons in order to keep it simple. The instances of Keyshape belonging to one particular appearance of an object are collected in a Keyshapes entity, which also specifies the type of the shapes: rectangle or oval. Each Keyshapes is again related to an Interval, and thus to one or more ontology concepts. The relationship between Interval and Keyshapes is one-to-one, so they could be combined to simplify the implementation of the model.

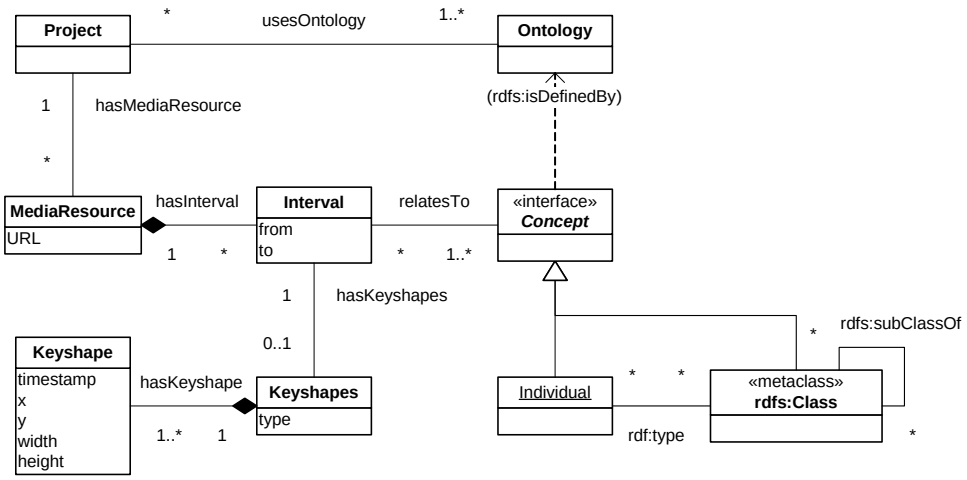


**Figure 6.8: OntoLog model with spatial extensions**

As an example of spatial annotation using this model, consider a shot where person A is standing in the middle of the screen, and person B comes walking in from the left, stops for a minute to greet person A before continuing out of the screen to the right. This would be represented as follows:

- Two individuals, A and B, both instances of a Person class.

- Two Intervals covering the entire shot – one related to A and the other to B.
- For person A: The Interval has a rectangular Keyshapes, containing a single Keyshape with appropriate coordinates. Just a single shape is needed, since A does not move.
- For person B: The Interval has a rectangular Keyshapes, containing four instances of Keyshape. All of them have roughly the same size and Y coordinate, since B moves pretty much straight across the screen, without changing size or shape. The first Keyshape has a timestamp corresponding to the beginning of the Interval, and a position to the far left of the screen. The second and third Keyshape are positioned at the same location – in the middle of the screen – but have different timestamps: The second is stamped at the time B reaches A, and the third when B leaves A again. The fourth has the end of the Interval as its timestamp, and is positioned at the far right of the screen.

Figure 6.9 shows the user interface of the proof-of-concept prototype implemented and integrated into the OntoLog application. The tree list on the left shows the ontologies, as usual. The example project in this case is the study of lobster larvae – larva mortality and cannibalism is a problem in lobster breeding, and marine biologists hope to understand the causes better by studying larva behaviour under various circumstances. "Aggressive larva B" is selected, and the video window in the middle shows the larva's rectangular bounding box. The list to the right of the video contains the keyshapes belonging to the selected interval, and the keyshape is edited by dragging it (or its corners) around on the video area.


**Figure 6.9: Prototype user interface for spatial annotations in OntoLog**

### 6.4.3 Discussion

The model described above is reasonably simple yet expressive. It meshes agreeably with the concept-based annotation scheme, and lends itself well to visualisation and manual annotation. It is expressive enough to be classified as "dynamic regions" according to the categories defined in section 3.1.2, but it also has the simplicity of the "static regions" scheme, if that is all that is required. Like the rest of the OntoLog model, it is able to manage annotation at various levels of detail – for more details, just add more keyshapes; for less detail, use fewer (and perhaps make them bigger to compensate).

As for the user interface, a small usability evaluation was performed, with two parts:

- A qualitative, formative part, designed to identify possibilities for improvement. This identified minor niggles with button design and use of colour, but the keyshape approach was well received.
- A quantitative, summative part, timing the production of spatial annotations. It was found that recording the movement of a single object took from 12 to 34 times as much time as the length of the interval(s) where it appears, depending on movement complexity and the user's experience level.

## 6.5 OntoLog Crawler – web-based browsing and searching

OntoLog Crawler is a web-based searching and browsing system for OntoLog data. An article describing it has been published [Austvik et al. 2003], included as appendix D. A fuller account of the work is available as a technical report [Austvik and Meland 2002]; it is summarised below.

### 6.5.1 Objectives and requirements

The main objective of OntoLog Crawler is to provide a search system, and a browsing system that is less limited than the intra-video browsing of the OntoLog application. The system should take advantage of users' familiarity with web browsers, hypertext documents and web-based search systems to design a consistent and user-friendly system. The network structure of the RDF model appears to be a good match for this approach.

The search system should take advantage of the semantics of the OntoLog model. When finding matches to a user's search terms, for instance, it should examine the types of the matching resources and use this information to present the result in the appropriate manner, and to expand the search according to the subclass and instance semantics of OntoLog's ontologies, if applicable.

When presenting media clips as the result of a query, the system should be able to handle overlapping intervals and concatenate disjoint segments, so as to present the entire result as a virtual edit of the underlying media material.

### 6.5.2 Design and implementation

The most basic functionality of OntoLog Crawler is its RDF browser, shown in Figure 6.10. This shows the characteristics of a given RDF resource[1], its URI and the statements it participates in, both as subject, predicate and object. The resources in each statement are hyperlinks to pages describing them in the same way; there are also links to the models and namespaces the resources belong to. OntoLog Crawler also provides an RDF search interface, where the user can find statements and resources matching some search term.

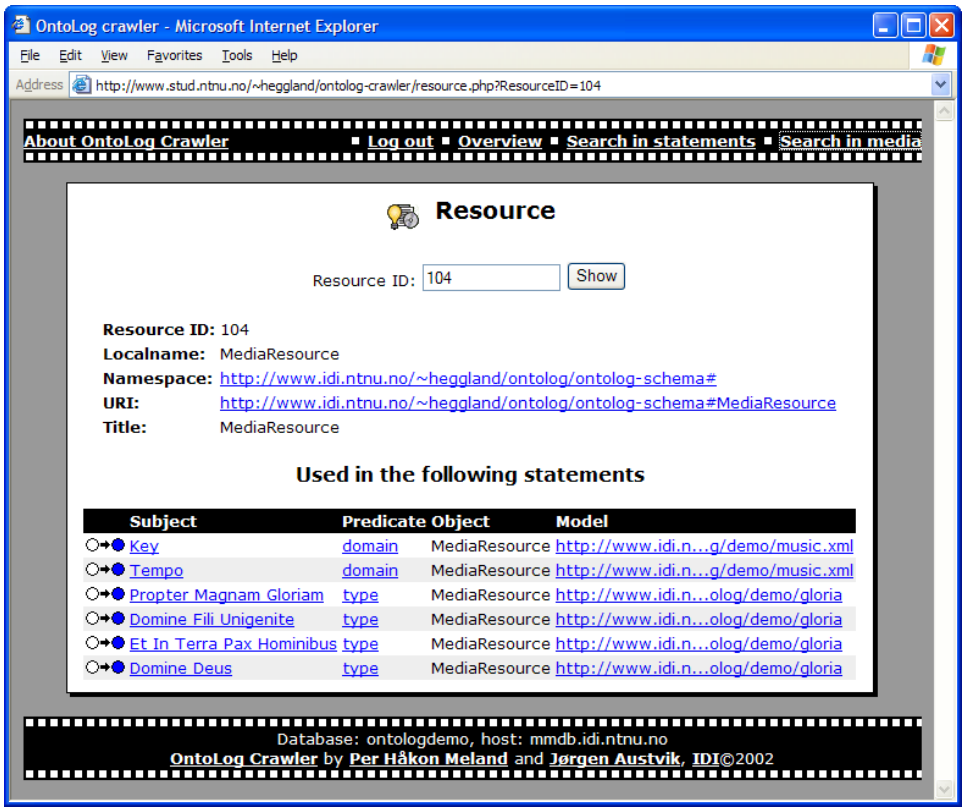


**Figure 6.10: OntoLog Crawler describing an RDF resource**

This is a quite low-level perspective on the data. It is excellent for exploring the inner workings of OntoLog (or other RDF-based systems), but you have to be very familiar with the RDF model and the way OntoLog uses it to make much sense of the data.

Another perspective is based on the semantics of OntoLog's conceptual model – projects containing media resources and ontologies, media resources containing intervals related to concepts in ontologies, and so on. OntoLog Crawler creates web pages describing these entities, with the relations between them as

[1] This particular screenshot describes the MediaResource resource, one of the classes in OntoLog's conceptual model (cf. section 5.1.1).

hyperlinks. For instance, Figure 6.11 shows the description of a concept: the soloist Guro Evensen, who is part of the Participants ontology. The page shows the properties describing her, the intervals she is connected to in various media resources, and her place in the ontology. The links in the Intervals table provide playback functionality, and the "Play merged resultset" lets you experience a virtual media document created by concatenating all the intervals. A similar function is available for subtrees in the ontology; you can for instance play a merging of all the intervals that are related to the concept Soloist or any of its descendants.

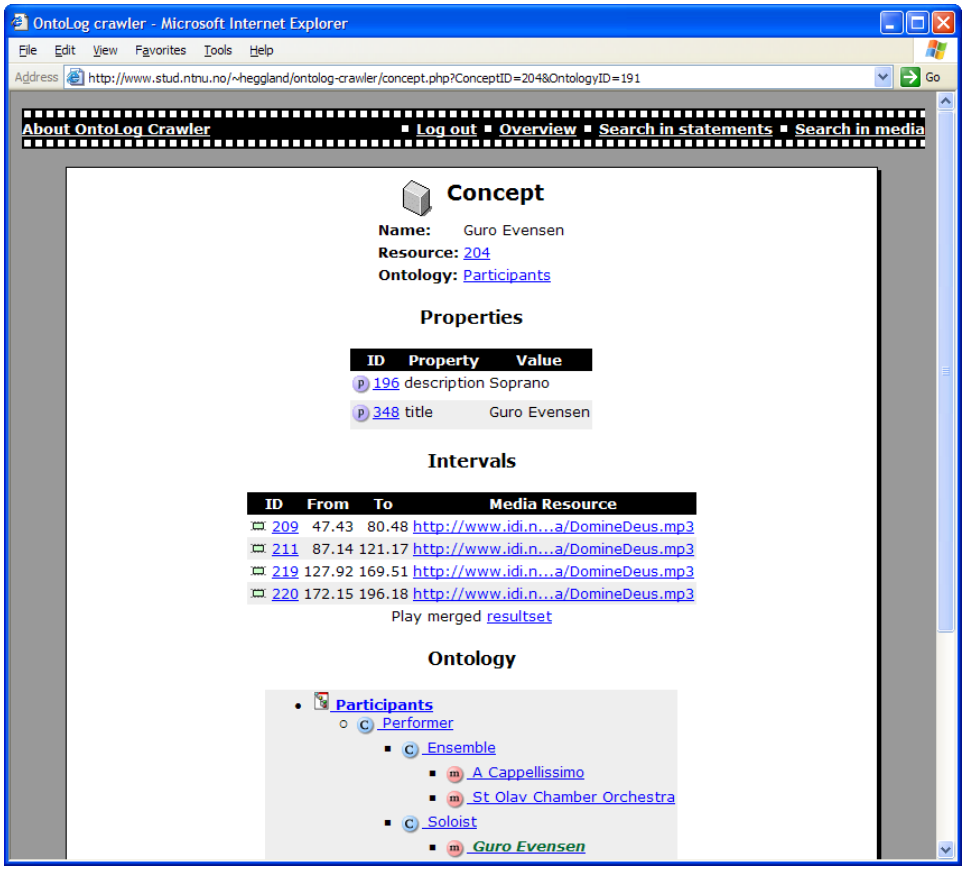


**Figure 6.11: OntoLog Crawler describing a concept**

Similar pages describe the other elements of OntoLog's conceptual model: projects, with lists of media resources, schemas and ontologies; ontologies, with their concepts organised hierarchically; and media resources, with lists of intervals.

A corresponding search system is also provided, shown in Figure 6.12. This example shows the result of a search after the term "solo". The system has performed an automatic query expansion: It has found only one resource matching "solo", namely the class "Soloist". This class is not directly related to any intervals, but it has two individuals that are: Jon Fredrik Hjemli and Guro

Evensen. Thus, these intervals are presented as the result. The media resources matching the search term (either through their own properties or through containing intervals that match) are also shown in their own table; last, but not least, the matching concepts are shown, in the context of the ontology they belong to.

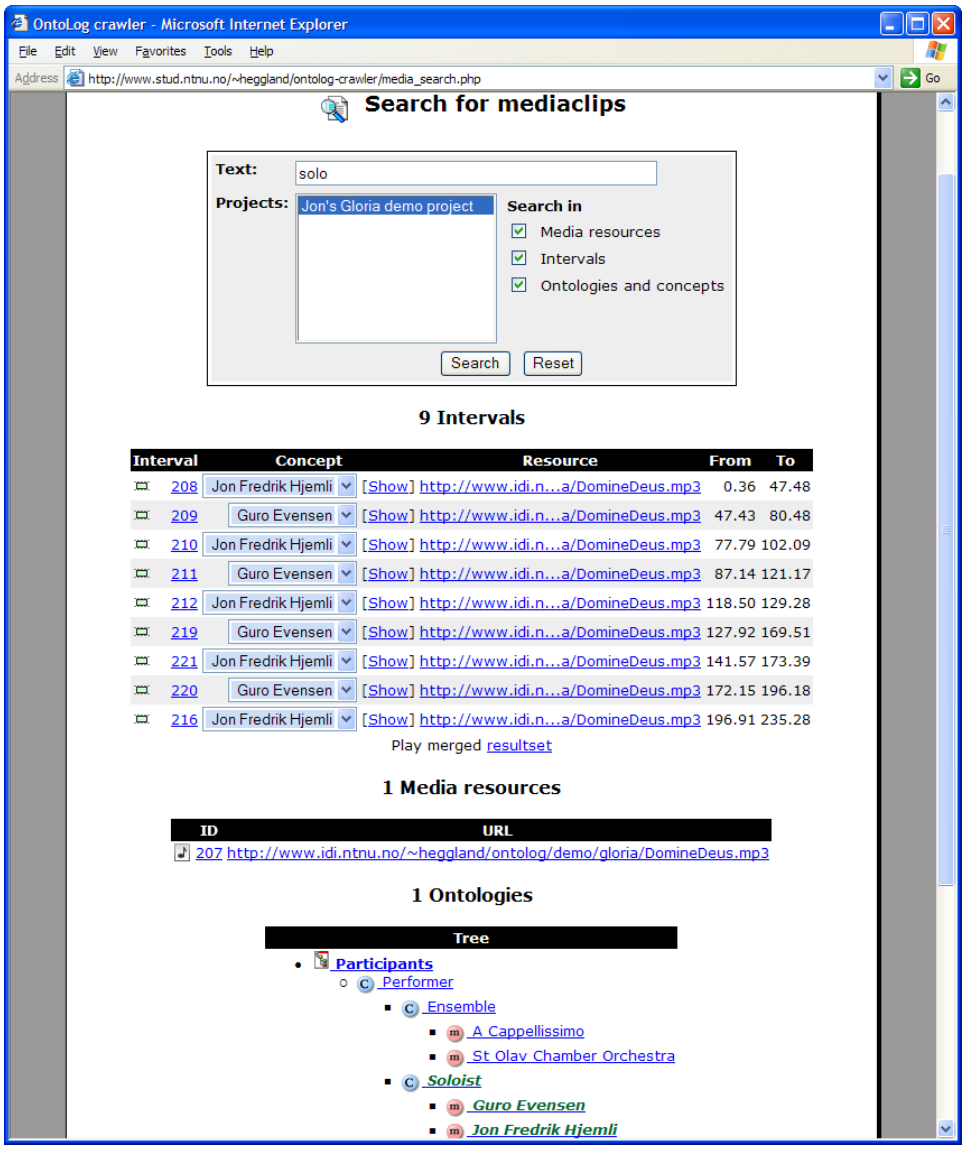


**Figure 6.12: OntoLog Crawler's media search**

### 6.5.3 Discussion

The hypertext paradigm is an obvious match for OntoLog's RDF-based annotation model. The graph structure of the data facilitates simple, uniform and powerful browsing capabilities, using an interaction technique familiar to most

users. The use of ontology semantics in processing search results is a great help for generating more meaningful results and presenting them in a structured manner.

A quantitative usability test showed that users are reasonably satisfied with OntoLog Crawler, though they find it somewhat complex to use. The hyperlink-based interface is consistent and simple, but the presentation of data and search results is at times confusing, and not very visually oriented – more often than not, the interface consists of tables of links.

OntoLog Crawler was excellent for advanced users with a thorough understanding of the RDF model and how it is used by OntoLog. However, another approach may be more appropriate for more casual or inexperienced users. This is explored further in sections 6.7, 6.8 and 6.9.

## 6.6 Ana – temporal algebra and annotation analysis

Concept-based temporal stratified semantic annotations create rich metadata sets, where much potentially interesting knowledge is implicit in the relations between different intervals and their related concepts. Ana is an analysis tool designed to unearth this information, to make it explicit. The design, implementation and evaluation of the tool are described in [Rystad 2002] as well as in this section.

### 6.6.1 Objectives and requirements

In many branches of knowledge work, e.g. anthropology, media and athletic science, video recordings are analysed to establish patterns of movement or behaviour by noting frequency, duration and sequences of activity. As an example, in the lecture database from sections 4.1.5 and 5.2.5, someone might want to know how often Examples (a concept) of Program Flow (another concept, with subconcepts) are interrupted by questions (a third concept). The objective of Ana is to facilitate this kind of analysis, exploiting the ontology-based stratified annotation scheme of OntoLog.

Given a set of intervals from a collection of media resources, it is possible to derive several statistics from them: number of intervals, average interval length, standard deviation, total length and so on. However, the perhaps most interesting analysis comes from looking at how intervals related to two or more concepts interact – how they overlap, which typically follows the other, and so on. This can be done by constructing expressions using temporal operators and relations between sets of intervals.

Temporal set operators create new intervals based on two sets of intervals, as illustrated in Figure 6.13. Based on these three operators – union, intersection and minus (subtraction), arbitrarily complex expressions can be formed.

The thirteen temporal relationships identified in [Allen 1983] are also useful for analysis. Seven of these are illustrated in Figure 6.14; the remaining six are reciprocals of those shown (except the “equals” relationship, which is symmetrical). These can be used to specify a subset of a set of intervals; for instance the intervals of set A that meet some interval in set B.

These operators are closed over the set of intervals – they take interval sets as input, and produce interval sets as output – and can therefore easily be concatenated into complex and precise expressions. Thus, the user can specify a particular set of intervals s/he is interested in, and perform a statistical analysis of it. Ana should provide the functionality for doing so, utilising the semantics of OntoLog's ontology-based stratification model.

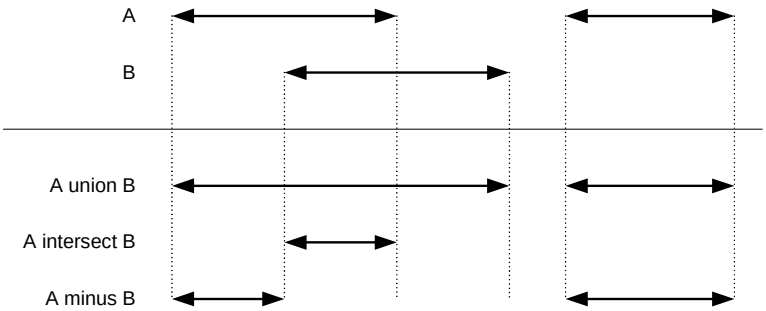


**Figure 6.13: Temporal set operations**

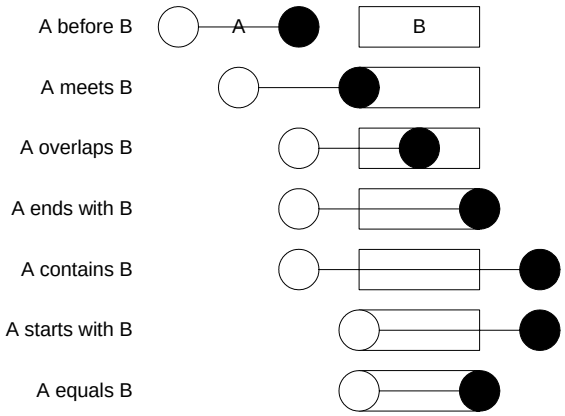


**Figure 6.14: Temporal relationships**

### 6.6.2 Design and implementation

The main idea of Ana is to construct temporal algebra trees based on the ontologies of a particular project. Figure 6.15 shows Ana's main window (Norwegian user interface). The tree list on the left shows the ontologies used by the selected projects, the concepts of which are used to construct the algebra tree on the right using the various controls. For this construction, Ana provides temporal relationship operators and set operators, as shown in Figure 6.16. Algebra trees of arbitrary complexity can be created.

The properties (the computed statistics) of the selected algebra node are shown in the table at the bottom of the main window. Any node can be selected, not just the top level one. In Figure 6.15, the lecture database example from sections 4.1.5 and 5.2.5 is being analysed, and the selected node represents the intervals where the lecturer is showing examples of program flow while being interrupted by

questions from the students – "(Program Flow intersect Example) contains Question".

Also here, the semantics of the ontology structure is utilised. The interval set represented by "Program Flow" includes all the intervals related to the subclasses of "Program Flow" – "Loops" and "if statements", for example. Thus, analysis can be performed at different levels of detail and specificness, through judicious selection of concepts.

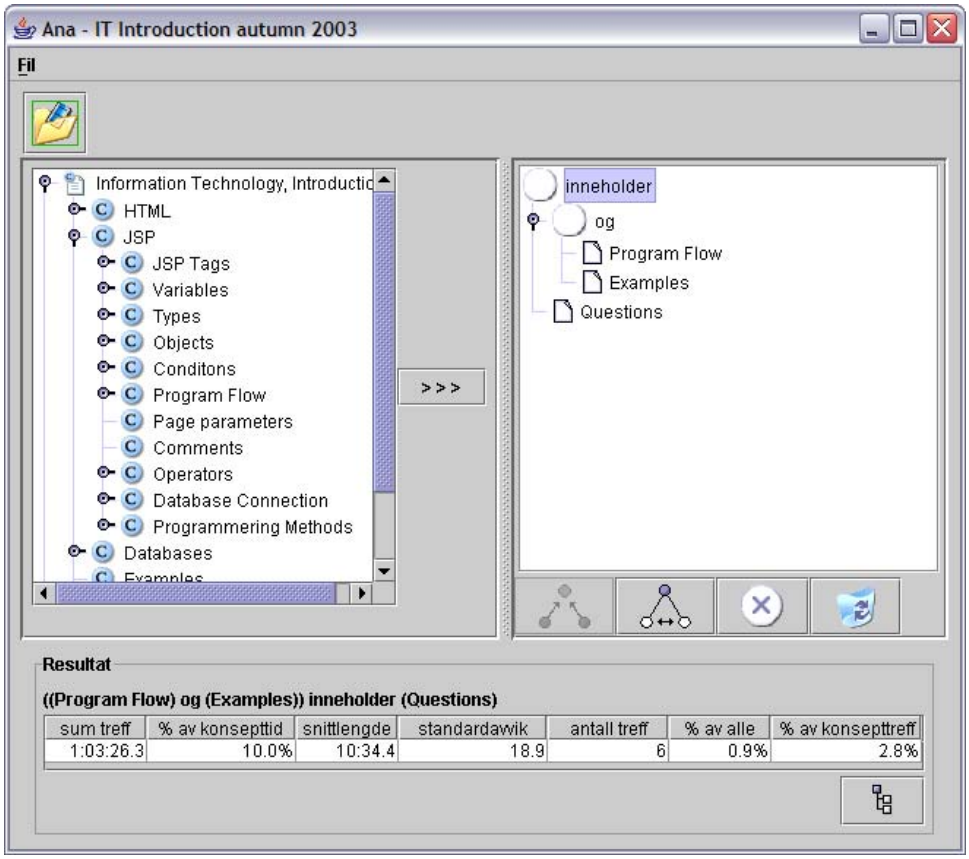


**Figure 6.15: Ana**

The properties computed from this interval set include among other things total amount of time, average length of the intervals and standard deviation of this average. Other measures are conceivable, and simple to implement. One could also easily imagine a visual display of the resulting interval set, and using this to access and control media playback – which would turn Ana into a query system as well as an analysis tool.

### 6.6.3 Discussion

Ana is a prototype made under time constraints, and suffers from several weak points in the graphical user interface, such as the multiple "OK" buttons in the temporal operator selection window in Figure 6.16. Nevertheless, it achieved

good results in a usability test, with regard to both ease of use, power, responsiveness and flexibility. Questions may be asked about the reliability of this test, though, as the test subject had nothing to compare Ana with.

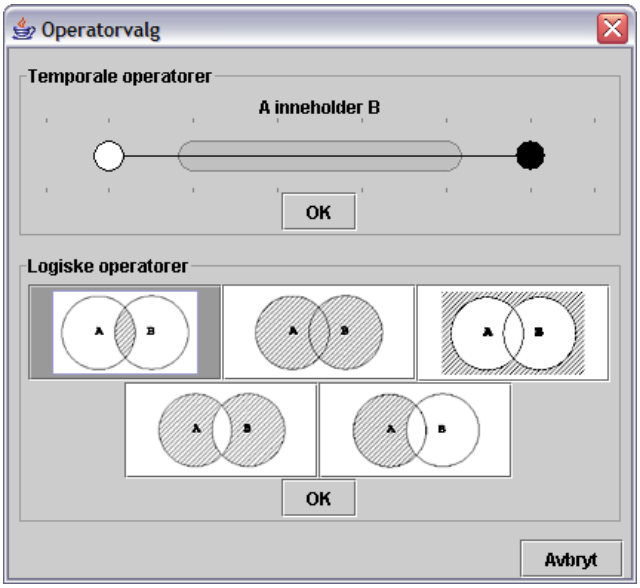


**Figure 6.16: Ana's operator selection window**

Performance was also tested, though not the performance of the temporal operators or the analysis algorithms. Ana reads all the concepts and intervals into memory when the media resources are selected, so the bottleneck was this loading process – the temporal algebra operations were as good as instantaneous. Testing the loading process with a large number of media resources, each with an average of 40 intervals, revealed that the system managed to load only one media resource per second, which is unreasonably slow. However, these disappointing results may be in part caused by the use of a beta release of the MySQL DBMS, as well as an early version of the Jena RDF library.

## 6.7 Savanta – Search, analysis, visualisation and navigation

Each of the tools described above have their strong points, as well as weaknesses. The ontology-based hierarchical interval visualisation of OntoLog is a good idea, but shows only one video at a time. OntoLog Crawler's web-based browsing and navigation is a popular interaction method, and its concatenation and presentation of media search results is interesting – but it is easy to get lost among the links, and hard to get a proper overview of the database contents. Ana has powerful mechanisms for selecting just the parts of the media that you are interested in, but has few ambitions when it comes to what to actually do with it.

It seems possible that a system could be designed and built that draws on the good ideas from all these systems – one that integrates visualisation, browsing, search, filtering and analysis, maximising the advantages and minimising the problems

described above. This section presents Savanta – Search, Analysis, Visualisation And Navigation in Temporal Annotations – an attempt at such a system.

*Note:* The "we" in the text relating to Savanta (and Savantoogle and Forms) refers to Jon Olav Hauglid as well as the present author; see appendix E.

### 6.7.1 Methods for Information Gathering

The content in a video database containing temporal annotations can be seen as a collection of media resources, each with a collection of temporal intervals described with metadata. This conceptual model is illustrated in Figure 6.17.

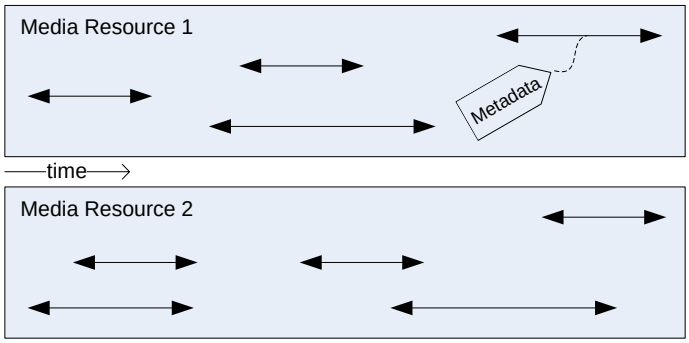


**Figure 6.17: Conceptual model of temporal annotation databases.**

In order for the user to see this information, some sort of visual presentation must be available. The process of creating such a presentation is called *information visualisation* – more formally defined as "the process of transforming data, information, and knowledge into visual form making use of humans' natural visual capabilities" [Gershon et al. 1998].

Depending on the size and complexity of the database in question, a large number of different presentations can be imagined – from the detailed display of a single piece of metadata to an overview of the whole database. Limiting a system to a single type of presentation would clearly be too restrictive. Some form of *navigation* between the different presentations is therefore required.

As the media database grows, it becomes necessary to be able to limit the presentation to a subset of the stored data. This is both because displaying everything would make the presentation too cluttered and because not every piece of information is equally relevant to a given user in a given setting. Generating a subset of the database can be done in a number of ways. The most common method is to use some sort of *query language* or *forms interface*. Based on one or more expressions entered by the user, a new ad-hoc collection of matching objects (or *query result*) is constructed.

*Filtering* is an alternative solution where the user removes uninteresting objects from the information collection by creating filters that eliminate objects that do not (or do) match certain criteria. A typical example is Dynamic Queries [Ahlberg et al. 1992], [Shneiderman 1996] where the filters are constructed by direct manipulation of interface widgets such as sliders or buttons.

A system for accessing temporal media databases needs not be limited to the data that is explicitly stored in the database. Due to the complex nature of temporal metadata, it may be useful to employ data mining techniques to derive new, high-level information that otherwise might be difficult and time-consuming to extract manually. This can include finding statistics such as the average length of registered intervals or identifying properties such as the most prevalent metadata in a given result. In addition to being valuable information in their own right, results of what can be dubbed a temporal *analysis* might also lend themselves to navigation and filtering actions, thereby providing a positive synergy. The rich nature of temporal media could very well make such derived information more important here than in traditional database settings.

An overview of a proposed system which integrates visualisation, navigation, search, filtering and analysis, and how they relate to each other, is shown in Figure 6.18.

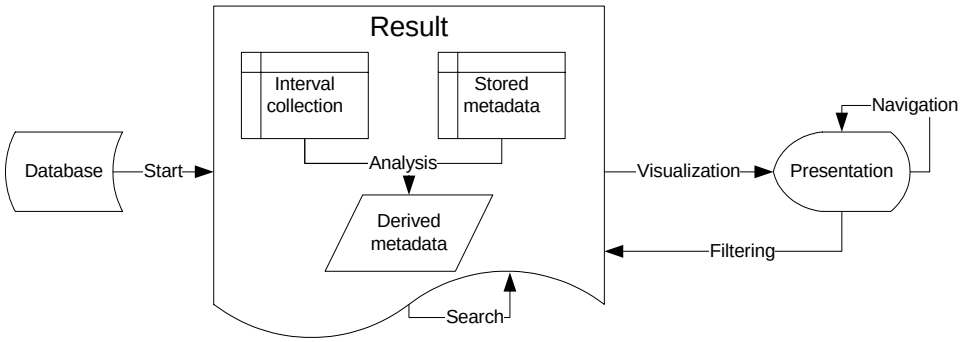


**Figure 6.18: Overview of a system for accessing data in a temporal media database.**

The terms used in the figure can be defined as follows:

- **Result**
  A collection of information objects (stored or derived) – could contain the whole database or just a subset.
- **Visualisation**
  The process of constructing a visual presentation of the result or a subset of the result. The presentation is displayed to the user in the user interface.
- **Navigation**
  Switching from one presentation to another.
- **Search**
  Producing a new result based on query expression entered by the user.
- **Filtering**
  Using the visual presentation of an information object as basis for the construction of a filter which removes uninteresting objects from the result.

- **Analysis**
  Using data mining techniques to dynamically derive new metadata based on information contained in the result.

We suggest that constructing a system which integrates many different methods for accessing information has a number of advantages. Simply because they represent different ways of making use of information, more power is available to the user. Information that might otherwise be difficult or even impossible to extract, can hopefully be made more accessible. A number of positive synergies can also be imagined. For example, the output of the analysis can be used for filtering – e.g. to remove intervals with the most recurrent male actor from the result. The analysis can also be made context sensitive with respect to the navigation choices of the user.

However, the integration of several methods will result in a more comprehensive interface which could become too complex and therefore difficult to use. In other words, the gain in power could be overridden by a loss of usability. This should be taken into consideration during design, and evaluated later.

### 6.7.2 Stored metadata

The "stored metadata" in Figure 6.18 is of course data structured according to the OntoLog model. However, this model is fairly complex; the ontologies in particular, with their classes, instances and properties, are potentially confusing for novice users. It is reasonable to target this kind of system at users who don't necessarily have a thorough understanding of conceptual modelling – it ought to be possible to use it as a retrieval interface for a digital video library without requiring a lot of training on the users' part. Therefore, we choose to present a simplified view of the model in Savanta:

- We don't consider the differences between classes and instances very significant, so we present them as the same thing, and call them *terms*. Thus, both the subclass-of relationship and the instance-of relationship are considered a narrower-term relationship.
- Likewise, we consider ontologies top-level terms. That they also define properties is not mentioned, though we of course still use the properties for describing things.
- We look at one project at a time. The media resources within a project are most likely closely related and homogeneously annotated – they are described using the same ontologies and terms. This will both simplify the model and better the chances of relevant analyses of the metadata.

This leads to the simplified view of the OntoLog model shown in Figure 6.19. Note that we do not actually change the model as such; we only change how it is presented to the user.

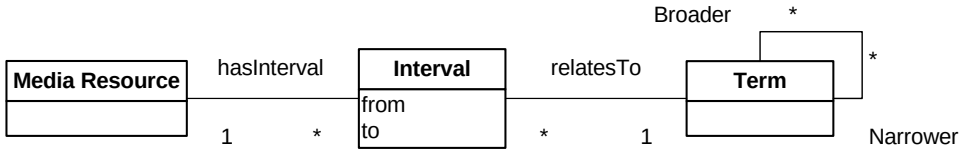


**Figure 6.19: Simplified annotation model**

This data is presented in Savanta as shown in Figure 6.20. Terms are presented in a tree list, with their associated intervals in a timeline display to the right. The terms, their "relatives" and their properties are also shown in a navigable hypertext panel at the right edge of the window. The user interface is further described in the subsequent sections.

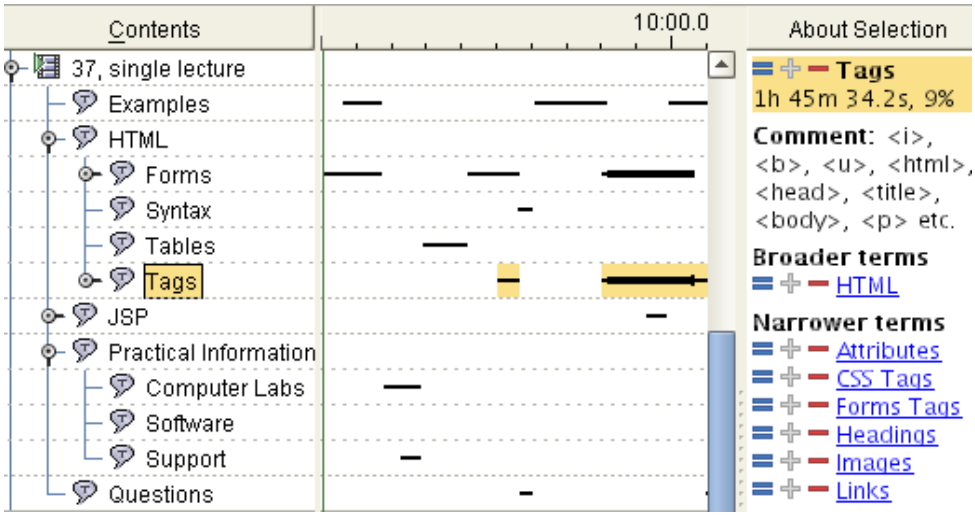


**Figure 6.20: Stored metadata in Savanta**

### 6.7.3 Derived metadata

A temporal annotation database contains much more information than what is explicitly registered. Consider a film clip where the intervals in which each actor appears, have been accurately registered. Implicitly, this database also contains information about the frequency of each actor and the temporal relationships between each of them. Examples of the latter might include actors that always appear together, actors that are always alone, etc. In order to extract such information, some sort of temporal analysis is necessary.

While intervals are the fundamental units in the model, semantic information is nearly always attached to sets of intervals – simply because annotating each separate interval is too time-consuming. As a result, a meaningful simplification is to derive information about sets of intervals rather than individual intervals. Four different, meaningful types of interval sets can be imagined in our system. The set of intervals, $T_{term}$, attached to a specific *term*, is perhaps the most important. Other sets include the whole database $D$, the currently displayed result $R$ and the set of intervals, $S$, selected by the user.

First of all, meaningful information can be derived about a single set of intervals. For the system described here, we have selected to display the total length of the intervals in *D, R* and *S*, both in seconds and relative to the total length of intervals in the database. As $T_{term}$ is a function of *term*, displaying such information about every (non-selected) term would simply be too overwhelming and confusing.

Temporal analysis gets much more interesting when we start to examine two sets of intervals and how they relate to each other. If we see an interval, and thus a set of intervals, as a set of points in time, we can use standard set operators as *temporal operators* to manipulate the intervals. This includes operators such as union ($\cup$), intersection ($\cap$) and subtraction ($-$).

Further, we can examine how two intervals, A and B, relate to each other on a common temporal axis. For example, A might have ended before B starts, they might be equal, or A might start exactly when B ends. Such temporal relationships were discussed in section 6.6.1. While temporal operators return one or two intervals, temporal relationships simply say whether a given relation exists or not between two intervals.

With a multitude of temporal operators and possible sets of intervals to look at, a large number of possible computations exist. It is therefore necessary to focus on what gives results that are meaningful to the user. Our novel idea is to use temporal analysis to identify interesting *terms* by examining $T_{term}$ in relation to *D, R* or *S* using temporal operators. The possibilities offered by this method, can be illustrated by a few examples:

- Identify terms that have most in common with the result – i.e. that have intervals overlapping as much as possible with the intervals in the result.
- Identify terms that have little or nothing in common with the result.
- Identify terms that have a lot in common with the selected intervals, but little in common with the result.
- Identify the media resources that topically are most similar to the intervals selected by the user.

These terms can represent interesting information in their own right as well as serve as input for filtering operations (example: remove all intervals attached to an identified term). For this implementation, we have chosen to group terms into three semantic categories. These three categories are:

- Described by
- Related to
- Differs from

An illustration of these three categories used in the following discussion is shown in Figure 6.21.

Please note that all interval sets can contain intervals from different media resources even if the figure above (for clarity) shows only one. For example, in a database containing media resources $M_1, M_2, \ldots, M_n$, $R$ will equal $R_{M1} \cup R_{M2} \cup \ldots$

$\cup R_{Mn}$. At any time when an operator is applied to two sets of intervals, intervals from a given media resource are handled separately. This means that $A \cup B$ is a shorthand for $\{A_{M1} \cup B_{M1}; A_{M2} \cup B_{M2}; \ldots ; A_{Mn} \cup B_{Mn}\}$.

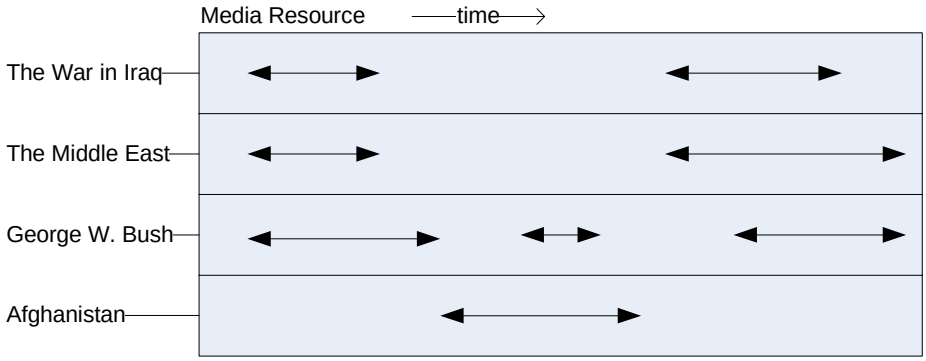


**Figure 6.21: The War in Iraq *described by* The Middle East; *related to* George W. Bush; *differs from* Afghanistan.**

### Described by

This category includes terms that should give a good description of the result or the currently selected intervals. To locate such terms, we identify terms where $T_{term}$ overlaps $R$ (or $R \cap S$) to as large extent as possible. In Figure 6.21, "The Middle East" intervals overlap all "The War in Iraq" intervals: "The Middle East" is therefore a good candidate for this category. The reasoning is that the degree in which two sets of intervals are equal, closely matches their semantic relation and thus one can be used to describe the other.

### Related to

"Related to" encompasses terms that are somewhat related to the result or the currently selected intervals. In practice, this means terms which have intervals that overlaps $R$ (or $R \cap S$) as close to 50 % as possible. For example, the "George W. Bush" intervals in Figure 6.21 have an approximately 50 % overlap with the "The War in Iraq" intervals, and they are therefore considered "related". The main purpose of this category is to serve as source for filtering operations. For this use, it makes sense to find ways of reducing the result by 50 % [Hauglid and Midtstraum 2002].

### Differs from

To describe an object fully, you not only need to include its properties, but also the properties it does not exhibit. This is covered by this category. In our case, it contains terms that have little or nothing in common with R (or R ∩ S) – that is, as little overlap as possible. In the example shown in Figure 6.21, the "Afghanistan" interval does not overlap any of the "The War in Iraq" intervals and thus is completely different. This category is also helpful for filtering, as described in section 0.

**Utility functions**

To find how well a given term is suited to each of the three categories mentioned above, we employ a set of *utility functions*. They take $T_{term}$ as argument and give a result from 0 (no utility) to 1 (high utility) which should indicate the usefulness of *term* with respect to a given category. The utility functions for each of the categories are given below.

*Described by (term):*

$$\frac{\sum_{\forall M} \left| T_{term_M} \cap R_M \right|}{\sum_{\forall M} \left| R_M \right|}$$

This is the length of the overlap between $T_{term}$ and $R$ totalled for all media resources, divided by the total length of $R$.

*Related to (term):*

$$4 \times \frac{\sum_{\forall M} \left| T_{term_M} \cap R_M \right|}{\sum_{\forall M} \left| R_M \right|} \times \left( 1 - \frac{\sum_{\forall M} \left| T_{term_M} \cap R_M \right|}{\sum_{\forall M} \left| R_M \right|} \right)$$

This is 4 × z × (1 - z) where z is the "described by" utility function. This gives a parabola function where the utility is maximised for 50 % overlap between $T_{term}$ and $R$ and minimised for 0 % and 100 % overlap. The scaling factor 4 ensures that the maximum value is 1.

*Differs from (term):*

$$\frac{\sum_{\forall M} \left| T_{term_M} - R_M \right|}{\sum_{\forall M} \left| M - R_M \right|} \times \left( 1 - \frac{\sum_{\forall M} \left| T_{term_M} \cap R_M \right|}{\sum_{\forall M} \left| R_M \right|} \right)$$

This function consists of two parts. The first is the total length of the intervals in $T_{term}$ which are not in $R$, divided by the length of the intervals not in $R$. This promotes terms with $T_{term}$ containing as much of what is not in $R$ as possible. The second part is 1 minus the "described by" utility function. This promotes terms with $T_{term}$ containing as little of $R$ as possible.

### 6.7.4 Visualisation

The Shneiderman mantra of "overview, zoom and filter, then details on demand" [Shneiderman 1997] is a good rule of thumb when designing interaction systems, and the process of visualisation ties into this in several ways. The system must construct the overview, produce the details on demand, and provide an environment in which the processes of zooming and filtering can be performed efficiently and intuitively.

The main challenge of creating an overview is the amount of data that must be presented. During the development and testing of this system, we created a sample database containing ten weeks of lectures in an entry-level computer science course; this amounts to 20 media resources, over 150 terms and over 500 intervals. While not much in terms of bytes, this is much information for a human to assimilate quickly, especially since much of the useful information is given by (possibly implicit) relationships – which terms are "active" at the same time as others, and how they are connected in the broader/narrower term hierarchy.

We use an extension of the technique presented in section 6.2.2 – timelines with interval lines aggregated according to the state of a term tree list. We use the media resources as top level nodes in the tree list, and present the term tree under each media resource node. Initially, all the nodes are collapsed, and this creates a representation compact enough to present a fairly big database on a single screen, as shown in Figure 6.22 (except that there, a single media resource node is expanded).

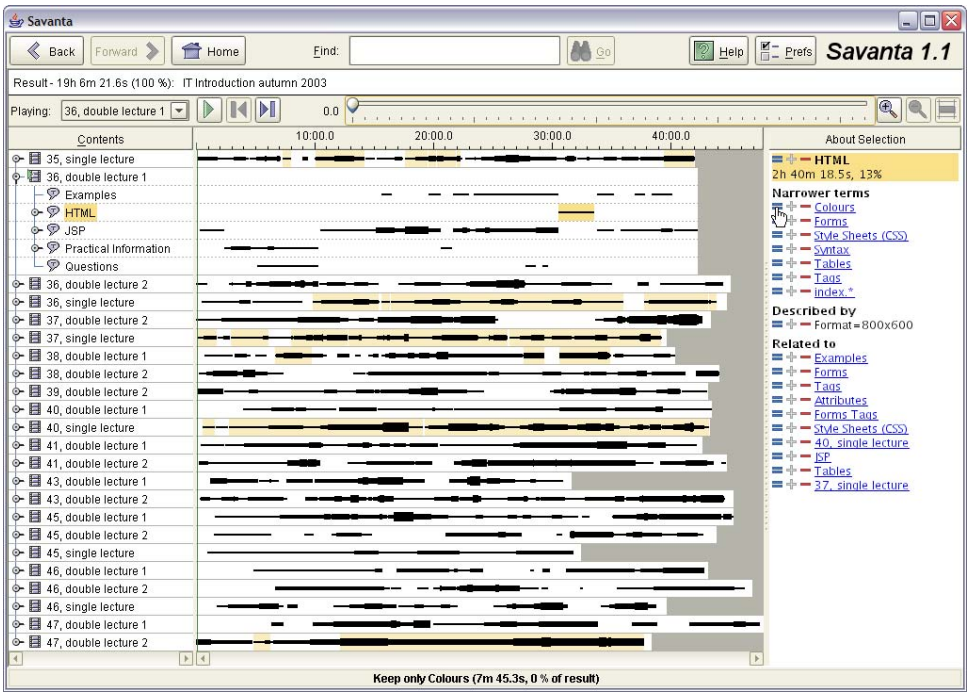


**Figure 6.22: Savanta**

Terse as this representation is, it is still possible to extract useful information from it. You get a list of the media resources in the database, and their relative lengths are readily apparent. It is also easy to see which media resources are heavily annotated and which are not, by noting the number and thicknesses of the interval lines, and where apparently nothing interesting happens, judging by the gaps in parts of the timeline.

"Details on demand" is handled in several ways in this visualisation. Selecting an item (a media resource or a term) displays information about it in the hypertext

panel to the right. This includes both stored and derived metadata. The stored metadata consists of the term's (or media resource's) properties, comment, and broader and narrower terms (if applicable). The derived metadata includes the length of the current selection, as well as the terms, properties and media resources related to the current result/selection, as discussed in the previous section. The hypertext panel also provides controls for navigation and filtering; more on this in the next sections.

Selecting a term also highlights the intervals it is related to in the interval display. Figure 6.22 also illustrates this, with the term "HTML" selected. Thus, it is easy for a user to at a glance find out when and how much a term is used.

Another mechanism for providing details (or "zooming"), is to expand and "drill down" in the tree list. Due to the semantics of the term hierarchy and the aggregation of related intervals, the visualisation at each level of the tree is useful: Even if no intervals are related directly to a given term, the intervals related to all of its narrower terms are displayed. If you want more details – exactly what aspects of JSP (Java Server Pages) are discussed in different parts of the expanded media resource in Figure 6.22, for instance – you simply expand the node. To avoid unnecessary clutter, only terms that are actually used in a media resource (or whose narrower terms are used) are displayed in the tree; but if desired, all the defined terms can be shown for completeness.

### 6.7.5 Navigation

*Navigation* is defined in [Baeza-Yates and Ribeiro-Neto 1999] as "following a chain of links, switching from one view to another, toward some goal, in a sequence of scan and select operations". Savanta's navigation capabilities are designed to offer the user relevant, context-sensitive paths through the information jungle. The selection, filtering and data mining mechanisms of Savanta directly affect and interplay with the navigation system.

For navigation through a media file, Savanta offers the traditional play/pause button and time slider, illustrated in Figure 6.23. In addition, two skip buttons are keyed to the currently selected part of the current result ($S \cap R$ or R, depending on whether anything is selected), so that navigating to an interesting media interval is a one-click operation. As default, only the current result (R) can be played; in this way is data irrelevant to the user simply and unobtrusively hidden.

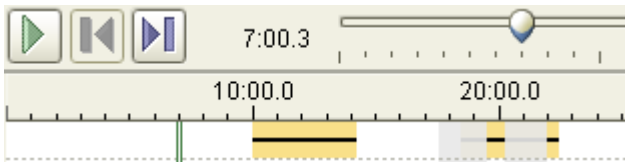


**Figure 6.23: Media navigation controls**

Navigating the collection of terms and media resources can be done by manipulating the tree list on the left. This is a common and intuitive method for displaying hierarchical data, familiar from e.g. Windows Explorer. This also

affects the configuration of the interval display – expanding nodes shows more detail, while collapsing them gives a greater overview.

An alternative method is to use the hypertext panel on the right, shown in Figure 6.24. This displays the selected term (or media resource) within its immediate context – narrower and broader terms – and among other things also shows the terms and media resources related to the selection, according to context-sensitive temporal analysis. All the items are clickable, so the user can easily browse both the term hierarchy as well as items related to the current result – thus creating dynamic paths through the material relevant to him/her. The goal the user is navigating toward can be different things – it could be simply to see the description and context of a particular term; or to visualise where various terms are used in the video corpus, and play selected intervals; or it could be to create filters, as discussed below.


**Figure 6.24: Hypertext navigation panel**

### 6.7.6 Filtering

As the default presentation in Savanta includes all the registered intervals in the database, it often contains items of little interest to the user. In order to make it possible to focus on relevant pieces of information, some means of eliminating unwanted items are required. In Savanta this can be done either by *filtering* or *searching*. These techniques differ in that filters are constructed by direct manipulation using interface gadgets, similar to the dynamic query approach [Ahlberg et al. 1992] [Shneiderman 1996], while searching require the user to input a textual query.

A filter, once constructed, is a set of intervals that change the subset of the database displayed to the user (i.e. updates the result). In theory, a filter could

contain a set of arbitrary intervals. In practice, we have limited the filters to contain intervals connected to a term or a media resource. This makes sense as terms and media resources are the basic semantic units. In addition, these units are also the output of the temporal analysis. A positive synergy between data mining and filtering can therefore be achieved. In this way, a user can construct a filter simply by selecting a term or media resource.

As both results ($R$) and filters ($F$) are sets of intervals, we can also here use set operations. Traditionally, a filter eliminates items from a result, that is $R \leftarrow R - F$. This requires the users to select what they wish to remove. However, previous experiences [Hauglid and Midtstraum 2002] indicate that users, given the choice, are more likely to construct filters based on what they wish to *keep* rather than remove. This suggests that it also should be possible to make filters that retain only what is common to both the result and the filter, or $R \leftarrow R \cap F$. Finally, to alleviate the problem of getting stuck in "local maximums", that is, removing relevant objects from the result, we have also made it possible to make filters that add the contents of the filter to the result, or $R \leftarrow R \cup F$. (Note that this breaks with the normal semantics of what a filter is.)

These three filter operations are respectively called remove (-), retain (=) and add (+) in the user interface. Filters are constructed by selecting the appropriate button on the right side of the interface as illustrated in Figure 6.25. For example, clicking the red + button beside "Comments", will add everything about comments to the current result.

The filter buttons are disabled if the corresponding filter will not change the contents of the result. Interestingly, this in itself provides yet another way of gaining information about the result. For example, if an add-button for a given term is disabled, this means that the intervals connected to this term are already present in the result – that is, $T_{term} \subseteq R$.

Filters are visualised as "greyed-out" areas in the interval display[1], as illustrated in Figure 6.25. The skip buttons discussed in the Navigation section are aware of the filters, making it easy to navigate to unfiltered, selected intervals. By default, playback of the selected media clip automatically skips filtered intervals. Thus, playing the intervals related to a term (or boolean combination of terms) is a one-click operation.

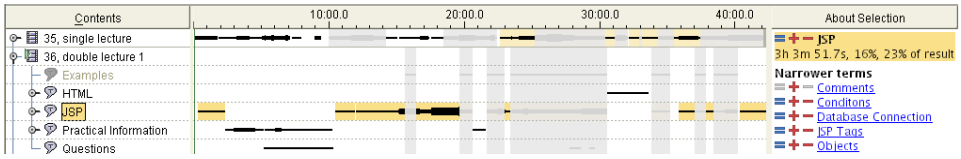


**Figure 6.25: Filters**

[1] Of course, "add" filters will decrease the "greyed-out" area, not increase it.

### 6.7.7 Searching

The searching part of Savanta allows the user to enter a textual query. This query is then matched against all textual attributes of terms and media resources. The intervals connected to these items are collected and used to modify the result. To retain the standard semantic of queries, the revised result is constructed as an intersection between the current result and the set of matched intervals. The interface components related to searching are visible at the top of Figure 6.22.

### 6.7.8 Discussion

Savanta is in many ways the most significant and original of the many tools presented in this chapter. Therefore, we though it appropriate to perform a proper formal usability evaluation of it; this is presented in the next chapter. For reasons explained there, we performed a comparative evaluation, comparing Savanta to two other searching and browsing interfaces based on more conventional paradigms. These are described in the two final subsections of this chapter.

## 6.8 Savantoogle – Google-like search

As discussed in the previous section, we built two systems comparable to Savanta in order to perform a formal evaluation of them. Savantoogle is one of these systems.

### 6.8.1 Objectives and requirements

The objective of Savantoogle was to provide a user interface for searching and browsing a video data base that is more conventional than Savanta. One such paradigm is information retrieval as used by web search engines, where search terms entered into a single text field produce a ranked list of documents. This has been used in video databases by e.g. SCAN [Whittaker et al. 1999], InforMedia [Christel and Martin 1998] and VoiceGraph [Oard 1997].

### 6.8.2 Design and implementation

Savantoogle, like its name suggests, is supposed to resemble Google[1] and similar search engines in looks and functionality. These search engines (Google, AltaVista[2] and AllTheWeb[3] being among the most popular) are familiar to most Internet users today, and are very easy to use. They consist of a single text field where the user can enter one or more words or phrases; executing the query then produces a list of documents, ranked according to how well each document matched the word(s) entered by the user. In a multi-word query, words can be prefixed with a '+', denoting that the word must be present in the resulting

[1] http://www.google.com/

[2] http://www.altavista.com/

[3] http://www.alltheweb.com/

documents, or a '-', meaning that the word must not be found. Figure 6.26 shows Savantoogle.

In the web search engines, the documents comprising the result are web pages, and the search is based on the occurrence of words in each document. In Savantoogle, this is slightly different, with media resources taking the place of HTML documents. The query words are matched against the terms, the terms' properties and the media resource titles and comments for each media resource. The ranking is mainly based on how much time is spent on each matching term – analogous to how web search engines rank documents according to the number of occurrences of the search terms. Media resources matching several or all the words in a multi-word query are ranked higher than those matching only a few. Matches in media resources titles or comments also increase the rank; this corresponds to how web search engines consider matches found in headings more important.

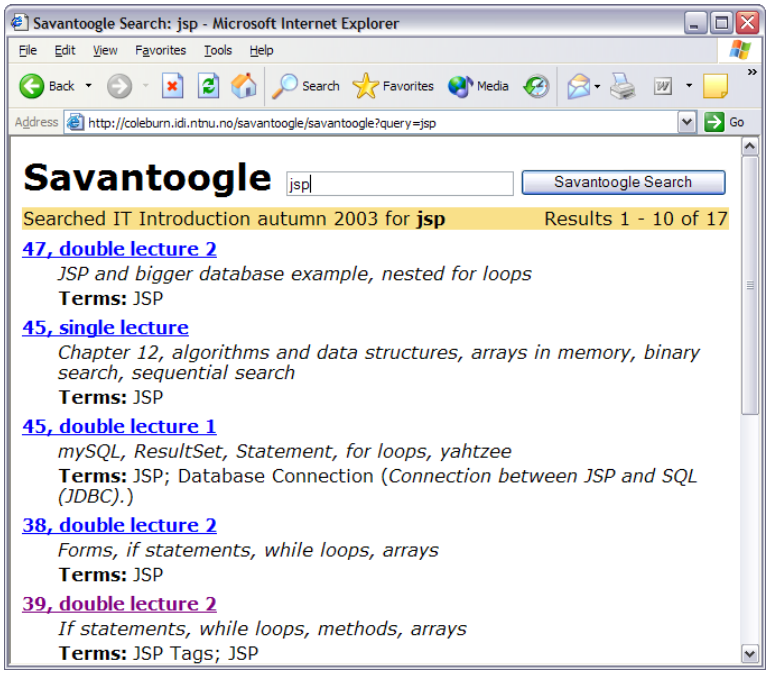


**Figure 6.26: Savantoogle**

Clicking a search result link (which in the web search engines brings you to the corresponding web page) brings you to an applet visualising the annotations belonging to the media resource in question, shown in Figure 6.27. This resembles the term tree and interval display of Savanta, but has noticeably less functionality. It is meant to correspond to the plain display of a web page in the web search engines, and to the relatively simple video browsing functionality of systems like Vane [Carrer et al. 1997] and Jabber [Kominek and Kazman 1997]. Most notably, the hierarchical aggregation of interval lines is not employed; instead, the term tree is initially completely expanded. Another limitation is the

fact that it shows only one media resource at a time. Video playback and skipping work like in Savanta, but there is no filtering or analysis functionality.

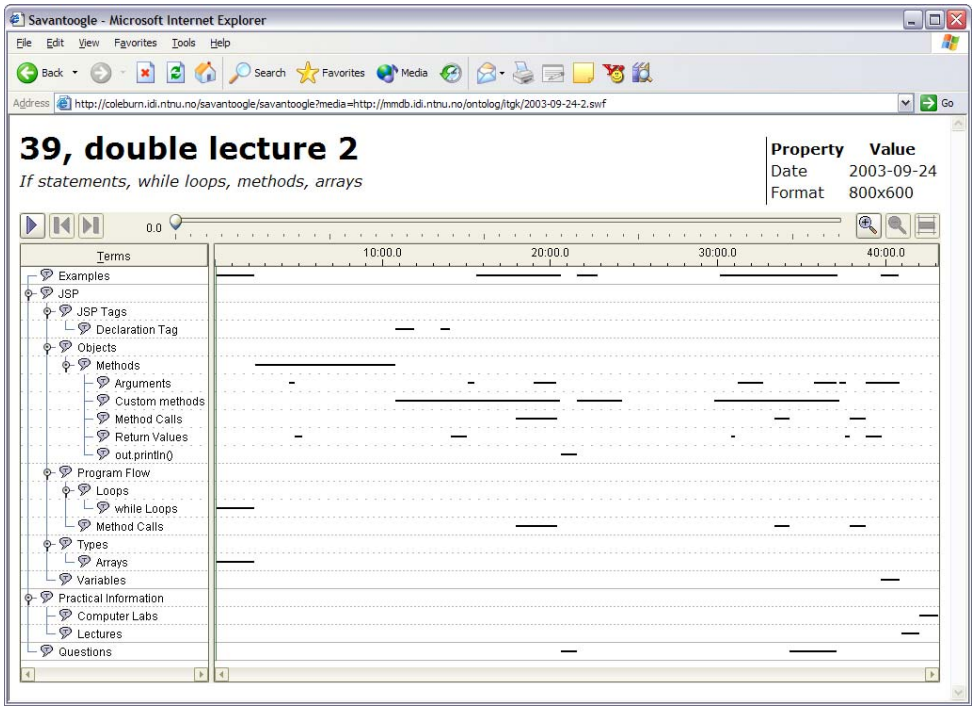


**Figure 6.27: Savantapplet, Savantoogle's interface for presenting a single media "document"**

Savantoogle's search engine is implemented as a Java servlet, while the media resource browser is (as mentioned) a Java applet. Both can be run from a web browser supporting Java 1.4, and both utilise the same basic data structures as Savanta.

### 6.8.3 Discussion

Savantoogle is evaluated formally, along with Savanta and Forms (see the next subsection) in the next chapter.

## 6.9 Forms – forms-based search

As mentioned earlier, we built two systems comparable to Savanta in order to perform a formal evaluation of them. Forms is the other of these two.

### 6.9.1 Objectives and requirements

The objective of Forms, like Savantoogle, was to provide a user interface for searching and browsing a video data base that is more conventional than Savanta. Many video database systems, such as OVID [Oomoto and Tanaka 1993], VideoSTAR [Hjelsvold and Midtstraum 1994], AVIS [Adali et al. 1996] and Algebraic Video [Weiss et al. 1995] define query languages for access to their

data structures. These are not very user-friendly, so graphical user interfaces are created that use forms to construct query expressions. The complexities of these interfaces vary, according to the complexity of the underlying models, but they use the same basic interaction model: select or enter query terms, use Boolean and/or temporal operators to combine them, and view the result in an unranked, textual list. Forms does this for the OntoLog/Savanta conceptual model.

### 6.9.2 Design and implementation

The Forms interface is shown in Figure 6.28. The terms are displayed in a tree list in the upper left of the window. By selecting term, and then pressing the "Select" button, they are copied across to the "Selected terms" list. Below this list, two radio buttons are used to switch between temporal 'and' (intersection) and temporal 'or' (union) of the query terms, when more than one term is selected. When the "Search" button is pressed, the query is performed, and the result is presented in the two-level tree list at the bottom of the window. The top-level nodes are the media resources (sorted lexicographically), and the leaf nodes are matching intervals within each media resource. No visualisation of the intervals is provided (other than the textual indications of start time, end time and length), since this is not common among the existing user interfaces based on this paradigm. The interface includes a simple video player in a separate window, with the same time slider and skip buttons as Savantoogle.

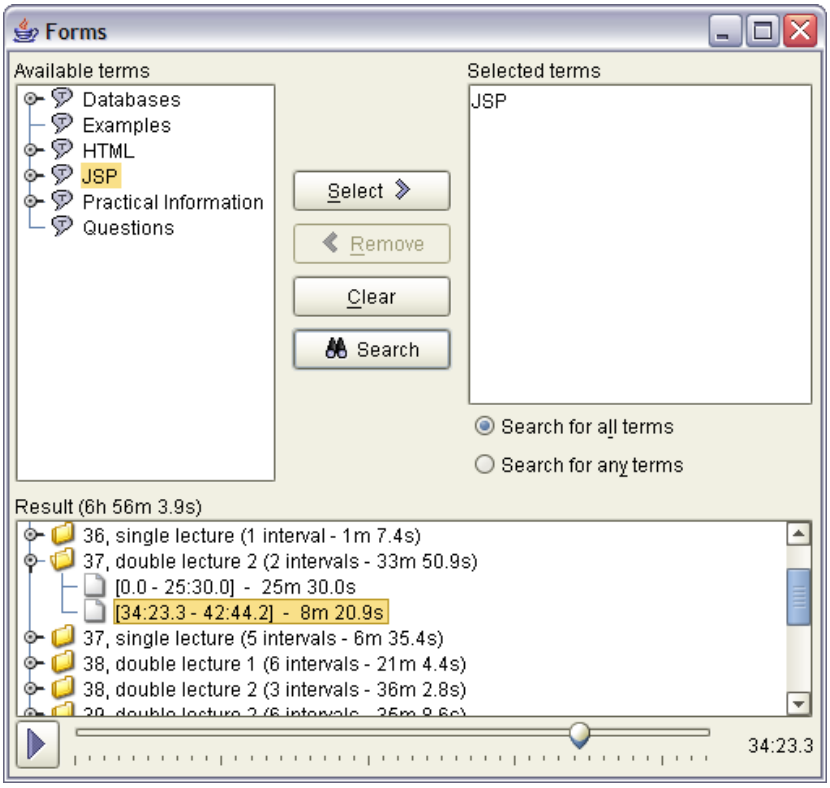


**Figure 6.28: Forms**

### 6.9.3 Discussion

Like Savantoogle, Forms is evaluated formally in the next chapter.

## 6.10 Summary

This chapter has presented various tools for managing and utilising OntoLog data: tools for manual and computer-assisted production of annotations and ontologies, for web-based browsing, searching and presentation, and for analysis using powerful and expressive temporal operators. The OntoLog model constitutes a strong foundation for such tools, and the next chapter investigates how these tools perform in practice.

# 7 The OntoLog system in the real world

This chapter consists of two parts. Section 7.1 presents the experiences gained from using the OntoLog system (and the OntoLog application in particular) for real-world purposes. By the nature of the system, it is quite difficult to perform a formal, quantitative evaluation of it; hence, I have used observation and informal interviews instead.

Section 7.2, on the other hand, presents a quantitative evaluation of Savanta. The searching, browsing and analysis tools are simpler to evaluate than the OntoLog application: They do not require much training (some are meant to be usable without training), and it is reasonably simple to find a group of fairly realistic users. Performing such evaluations is time-consuming, though. Because of this, and since most of the ideas explored in OntoLog, OntoLog Crawler and Ana are reused and refined in Savanta anyway, I have chosen to perform a full evaluation of Savanta only. However, since it is a comparative study, Savantoogle and Forms are also evaluated.

## 7.1 Practical use of the OntoLog system

Performing a quantitative evaluation of the OntoLog system as a whole, and of the OntoLog application in particular, would be very difficult. It is a fairly big system, and the OntoLog application requires quite a bit of training to use effectively. Thus, training users and performing formal tests would be prohibitively time-consuming.

It would also be hard to find enough appropriate test subjects. Creating temporal annotations with OntoLog is probably not something the casual computer-user would do; for realism, one would need to find a group of people that "are in the market" for such a system, so to speak. For reasons of validity, this group should be as homogeneous as possible, as well. Needless to say, this would be hard to pull off.

Thirdly, such a test would necessarily have to deal with a single scenario or annotation purpose (or at most a very small set), and would therefore not really evaluate one of the prime goals of the OntoLog system: the flexibility. It would also be a contrived situation – an annotation purpose that might not in fact reflect a real-world need.

For these reasons, I decided to do an evaluation in a different manner: To have OntoLog used for real-world tasks – that is, tasks that have a purpose beyond simply being a test case for OntoLog – by several different users, over a period of months or even years. During this period, I observed these users, got their input on problems and bugs, and improved the system according to their needs. At the end, I performed an informal interview with the users, to find out how well the OntoLog system performed in practice. Subsections 7.1.1 to 7.1.3 describe the three most important test cases – their context and purpose, the media material used, the ontologies that were developed, the production and use of the annotations, and a discussion of the lessons learnt. Subsection 7.1.4 collects the

experiences from a handful of less thorough (or successful) test cases, and subsection 7.1.5 summarises the section.

### 7.1.1 Evaluation of electronic medical records

Throughout the development of the OntoLog system, the doctoral studies of recent dr. med. Hallvard Lærum [Lærum 2004] have provided a crucial scenario and test case. Lærum has been the principal "expert reviewer" of OntoLog at the various stages of its design and implementation, and has provided a lot of useful suggestions, requirements and criticism.

#### Context and purpose

Lærum's task was to evaluate the use of electronic medical record (EMR) systems – to determine how and how much they were used, how well they worked in practice, and how they impacted on the routines of physicians, nurses and other medical personnel. To this end, he videotaped medical consultations, and annotated them in OntoLog. The original idea was to use these videos for "stimulated recall" interviews [Bloom 1953] with the physicians involved, but this stranded on account of the difficulty in recruiting a sufficient number of volunteers. However, OntoLog still contributed nicely to the task analysis necessary to design the questionnaires that in the end became the principal EMR evaluation method.

#### Media material

The media material used by Lærum was recordings of nine medical consultations, with a length ranging from 14 to 52 minutes. The total length of the material was four hours and 39 minutes. Other than the lengths (and the actual maladies afflicting the various patients), the videos were quite homogeneous; each was recorded as a single shot without any camera movement, showed a doctor interviewing a patient, and was not edited except for simple trimming of extraneous footage at the beginning and end.

#### Ontologies

Lærum constructed three ontologies, with different purposes and viewpoints. One was a single-level ontology of *Phases* – consisting of nine stages a typical consultation may go through: Preliminary work, Introduction, Anamnesis, Examination, Discussion and assessment, Direct remedial actions, Requisition of remedial actions, Summary and conclusion, and Complementary work. Additionally, three concepts were defined to classify intervals that were not related to a particular phase: Pause, Logistics and Unclassified.

The second was a more complex ontology of *Activities*, containing 36 more or less specific activities or tasks. This is shown in Figure 7.1.

The third, *Comments*, was a very small and simple ontology of two concepts: Doctor's comments and Observer's comments. These were used as a means for attaching comments and explanations to intervals of the video.

All the concepts are classes; these ontologies do not use individuals at all. This is primarily because they were originally designed at a time when the OntoLog model only supported one kind of hierarchical relation between concepts, but it is not likely that it would have been different even if individuals had been available. The concepts represent abstract tasks, activities and phases, and semantic difference between classes and individuals was not really applicable or needed by Lærum.

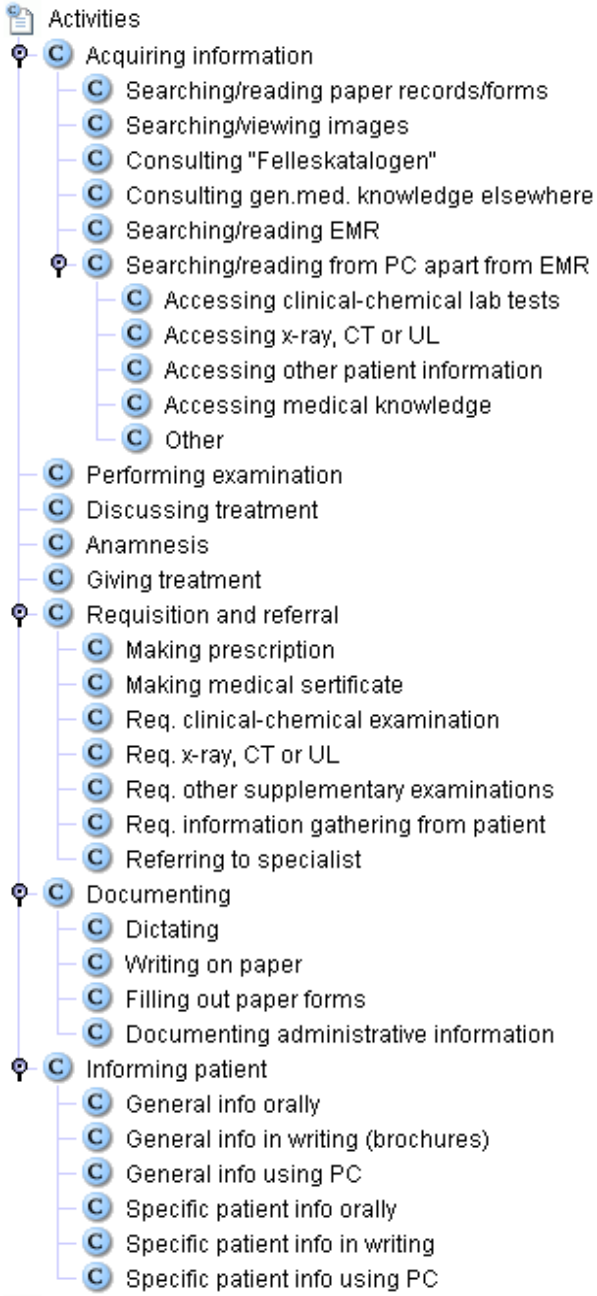


**Figure 7.1: Activities ontology (constructed by Lærum; my translation)**

### Production

The intervals and their related concepts comprise almost all the information in this project. About half of the concepts are described with a textual comment; the rest have nothing but a name. The media resources are not described at all, though this can be attributed to their small number – a larger collection would (according to Lærum) have to be described better.

The number of intervals per interview ranges from 42 to 135, with an average of 72, or around two intervals per minute. In the Phases ontology, the intervals do not overlap – they constitute a partition of the video – while in the Activities ontology, two intervals overlap about one third of the time. There is seldom more overlap than that. Most of the intervals are not described with anything, but in the Comments ontology, all the intervals have a comment property describing what goes on, e.g. "The e-mail system makes an annoying sound" or "Looking for the reflex hammer". These comments are the products of the stimulated recall methodology, where the subject and the observer review the recorded material, and the subject's actions and thought processes are explained and elaborated.

The logging was done manually, but the logger did not think of this as a hassle. The videos had to be watched carefully several times in any case, so using OntoLog to annotate them demanded little extra effort – indeed, using the annotations as an index made it significantly faster and simpler to navigate around in the video material while studying it. The logger commented that very detailed or accurate annotations would be more demanding to create, but given the material and the purpose of the annotation, he did not bother with better precision than plus/minus one second.

### Use

The annotations were used for various purposes. The segmentation provided by the Phases annotations functioned mainly as an index, to help the user navigate to particular parts in the video. There was an attempt to study the length of the different phases, with the purpose of determining if there was any correlation between phase length and the use of EMR systems, but this was abandoned: There is no universally accepted definition of such phases, so the results would have been too qualitative and dependent on personal judgment on where to say that one phase ends and another begins.

The Activities annotations were used for (and constructed as a part of) task analysis. An important part of the EMR project was to establish a suitable set of tasks that could be studied and evaluated relatively independently of the others, and OntoLog was used in the construction and validation of this set. Activity concepts were created, deleted, moved around, and intervals were subdivided and joined and moved from concept to concept as the most apt activity ontology emerged.

Last and least, the Comments annotations were used as comments. Events in the video needing special explanation, analysis or clarification were recorded as intervals related to one of the concepts in the ontology, and described with a short

text. Thus, OntoLog was used for three rather disparate purposes: as a segmentation-based video index / table of contents; for stratified, overlapping classification of the action in the films; and as a video notebook. An example is shown in Figure 7.2.

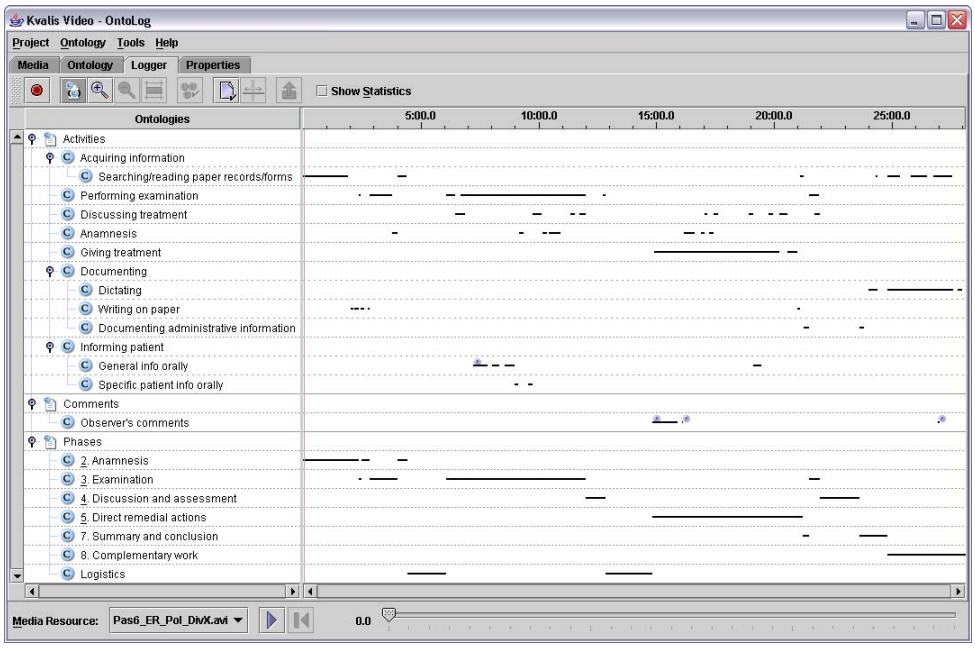


**Figure 7.2: OntoLog annotations for medical consultation**

## Discussion

OntoLog would have been a crucial part of the EMR project had it not been for the difficulty in finding volunteers for video observation. As it turned out, OntoLog was only moderately useful, but it was praised nevertheless.

The logger interface was especially commended. It was described as "the fastest method possible for manual annotation" due to its polished integration of timeline visualisation, ontologies, video navigation and interval editing. In fact, Lærum felt that the user interface "disappeared" – that he no longer interacted with buttons, lists and lines, but with video, concepts and intervals – which is the hallmark of a well-functioning tool. The keyboard-enabled video control and the direct-manipulation-style interval editing was particularly liked, and compared very favourably with QMA (see section 3.2.6), whose interval visualisation and interaction style was deemed quite inferior.

There were few shortcomings identified in the EMR project that were not remedied in subsequent iterations of the OntoLog implementation. The most important missing feature was the ability to create new video files based on the intervals related to a given concept (and its subconcepts): To "cut" all the relevant intervals from their respective films, and "paste" them together to create a new video. Aspects of this feature were implemented, though:

- The OntoLog application has a function for exporting the intervals related to a concept as a SMIL presentation, a textual file specifying to a SMIL-compatible media player which parts of a video (or collection of videos) it should play. Unfortunately, no SMIL player could handle the video format used in the EMR project.
- The search function in OntoLog Crawler lets you play the concatenation of the search result as a single virtual video. But again, this uses SMIL, and thus did not help the EMR project.
- Savanta allows you to filter away everything but a given concept (or arbitrary combination of concept, using temporal algebra), and can play only the unfiltered parts. Unfortunately, Savanta was not finished before the EMR project was.

Additionally, a need for analysis functions (like the ones QMA provide) was mentioned, but due to the way the video use in the project turned out, the requirements for this was never specified explicitly. Possibly the functionality of Ana or Savanta would have been sufficient.

### 7.1.2 Indexing lectures in information technology

In order to test OntoLog on a larger corpus of data, as well as provide a test case for the evaluation of Savanta in section 7.2, and provide a service to students, the lectures in the course "Information Technology, Introduction" at NTNU in the autumn semester of 2003 were recorded and annotated in OntoLog.

#### Context and purpose

The course "Information Technology, Introduction"[1] is taken in the first year by most (if not all) engineering students at the Norwegian University of Science and Technology (NTNU). Its syllabus is mainly programming with Java Server Pages (JSP), HTML and database technology (Entity-Relationship modelling and SQL), as well as some more theoretical informatics. The course consists of thirteen weeks of three 45-minute lectures per week. The course is lectured in a practical manner, with lots of examples of actual programming, so it is very suited for relatively detailed indexing – it is simple to ascertain exactly when while loops are being discussed and demonstrated, for instance, and it is useful to be able to review this demonstration at a later time.

The lectures were recorded and logged for several purposes:

- To create a relatively sizeable corpus of annotations, in order to test the OntoLog tools under reasonably realistic circumstances. 30-40 video files with a total length of 20-30 hours would hopefully expose some scalability problems in the user interfaces, and be a realistic application of the OntoLog system.

[1] http://itgk.idi.ntnu.no/

- To provide a service to the students taking the course. Many of them are already quite proficient in information technology, and skip the lectures. Recordings would make it possible for them to make sure they did not miss anything important. And of course, most students could benefit from reviewing the material while doing exercises or studying for the final examination.
- To provide a test case for the evaluation of Savanta (see section 7.2).

**Media material**

The media material is video lecture recordings, but it is not live video in the ordinary sense of the word. Rather, the screen image of the lecturer's PC has been recorded, which includes both his PowerPoint slides and his demonstrations of programming, HTML and database use. Consequently, the lecturer is never seen in the picture, but the audio track consists of his voice. This kind of recording was beneficial for several reasons:

- It was non-intrusive – no cameras or other obvious recording equipment was visible, which made it easier for the lecturer to relax and behave normally.
- No recording crew was needed – no cameraman needed to track the movement of the lecturer, for instance. All that was needed was to start recording at the beginning of the lecture, and stop it at the end.
- The video files could be compressed a great deal, even with lossless compression. A typical lecture is 43 minutes long and recorded with a resolution of 800x600, but is typically less than seven megabytes in size (with a rate of five frames per second).

27 lectures were recorded; some were missed due to technical problems. The total length of the material is a little over nineteen hours. The length of each lecture ranges from 32 to 48 minutes, but almost all are between 39 and 44. They are pretty uniform in structure – PowerPoint slides interspersed with practical examples and demonstrations – though a few consist only of slides.

**Ontologies**

The lectures are annotated with a single ontology, which consequently is rather large. It is shown partially expanded in Figure 7.3. It is quite detailed; it contains concepts for individual HTML tags and JSP types, for instance. 132 concepts are defined, but the ontology appears even larger, since a handful of the classes have multiple parents. For example, "Forms Tags" is a subclass of both "Forms" and "Tags".

Three of the six top-level classes are the main topics of the course: HTML, JSP and databases. The other three represent common events or activities in a lecture: Examples, Questions (from the students) and Practical Information (information that is does not pertain to the course syllabus, but is nevertheless relevant – information about exercises, support, computer labs, software and so on). This

subdivision suggests perhaps that using two distinct ontologies would have been more appropriate, but a single ontology was used for reasons of simplicity.

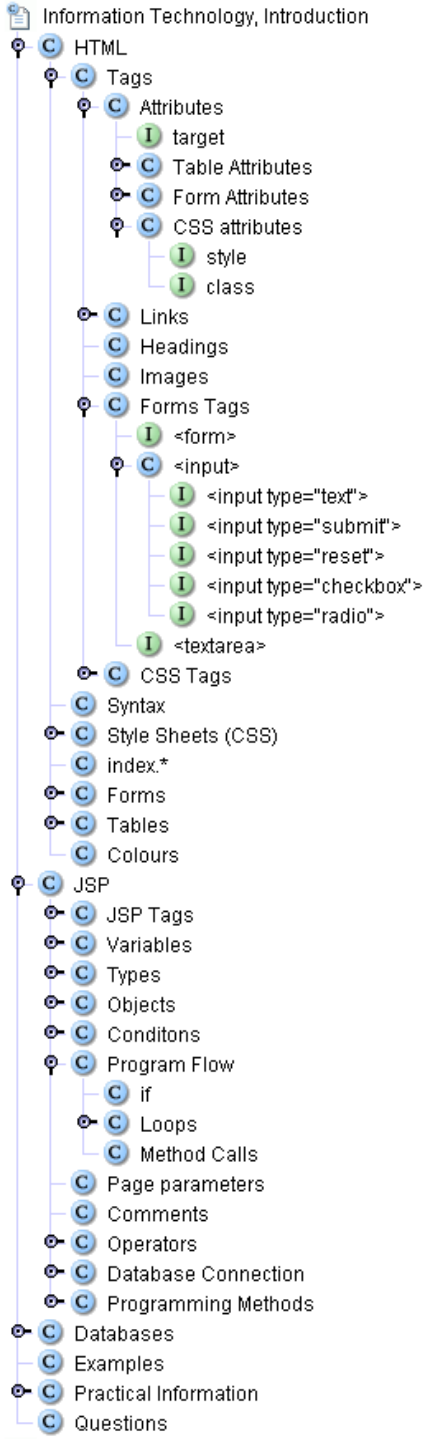


**Figure 7.3: Partially expanded ontology for the course "Information Technology, Introduction" (constructed by the present author)**

The ontology has a maximum depth of five, and a maximum fan-out (number of direct subclasses or individuals to any class) of twelve. Most of the classes have fewer than eight children. The concepts represent abstract topics or activities, not real-world objects (or classes of objects); however, the semantics of the class/individual distinction and the relationships between them are in some cases used "correctly": For instance, "<form>" is an individual of the "Forms Tags" class, which again is a subclass of the "Tags" class. This did not matter much in practice, though.

The ontology is tailored to the IT Introduction course, and as such quite specialised. Different courses would need different ontologies, though parts of the ontology might be reused. Courses dealing with HTML would probably find the "HTML" subtree useful; database courses might want to reuse the "Database" subtree and so on – and the three "lecture activity concepts" (Examples, Practical Information and Questions) are probably relevant for all courses. This is another point in favour of dividing this somewhat monolithic ontology into smaller, more cohesive parts, except that some classes are subclasses of classes in different subtrees – "Database Connection" is placed in both the "Database" and the "JSP" subtree. In any case, the ontology is general enough to be used by all the lectures in the course (though none of them uses more than about thirty of the concepts), and probably for subsequent runs of the course as well.

### Production

About a third of the concepts were described with a comment property – usually elaborating its definition, or referring to the pages in the course textbooks dealing with the concept. Each lecture was described with a one-sentence description of its contents as the comment property; additionally, the Dublin Core *date* and *format* properties were used to record the date of the lecture and the screen dimension of the video, respectively. Most of the lectures were recorded in 800x600, as previously mentioned, but some were 1024x768. No custom properties were used; one was defined – a *lecturer* property, with Media Resource as domain – but became rather irrelevant when it became clear that all the lectures would be given by the same lecturer. The project itself is annotated with a descriptive comment, as well as the Dublin Core *creator*, *contributor*, and *language* properties.

23 of the 27 lecture recordings are annotated with intervals. The remaining four dealt with theoretical topics that were not mentioned in any other lectures, and had a very simple structure (often just 4-5 PowerPoint slides) so it was not considered sufficiently useful to annotate them. The 23 temporally annotated lectures were described with 648 intervals in total; an average of almost 30, and a range of 8 to 46. There is temporal overlap between concepts more than half the time, and at the most "busy" times, seven concepts overlap. The longest intervals are about fifteen minutes, and the shortest just over ten seconds. Figure 6.22 in the previous chapter shows this project in Savanta.

The manual logging process was estimated to take about 50 % longer than the length of the lectures. A lot of this time was spent building and rearranging the

ontology; due to the relatively low number of intervals per lecture, the logger rarely had to "rewind" the video much or make more than one pass through it. Given a pre-existing, well-designed ontology, the logging time could most likely be reduced to just a few percent longer than the playback time. Note, however, that I did the logging myself, so I had of course intimate knowledge of and unmatched experience with the logging interface.

Automatic analysis for segmentation or classification should be relatively simple to perform for this kind of video. While there are no "shots" as such, there are very clean transitions between slides and applications such as text editors and (to a slightly lesser degree) between different applications, and the video images are crisp and practically noise-free. Optical character recognition should for the same reasons be relatively simple to perform.

**Use**

As previously mentioned, the annotations were produced for several reasons. As a test case for the evaluation of Savanta, they functioned very well: They provided a realistic body of data with enough richness to test and evaluate both simple queries, complex queries and exploration of OntoLog data, as described in section 7.2.

The project (or to be more precise, a version of Savanta configured to load the project) was also published on the World Wide Web, and demonstrated to the students taking the course. This proved a success; the web page of the project received over 600 unique visitors during the first two months after the presentation, and students characterised the system as "impressive", "extremely useful" and "should be used in more courses". However, interviews with students indicate that they mostly used Savanta and the annotations as a fairly simple index; the query, filtering and analysis functionality was for the most part ignored.

Finally, the project functioned as test data for some user interface issues (concerning scalability and performance, among other things), as discussed in the next section.

**Discussion**

The size of the project – the 23 videos, or 27 if you count the un-annotated ones – clearly exposed the browsing capabilities (not to mention the searching capabilities) of the OntoLog application as inadequate. This deficiency was remedied by Savanta, which presented an interactive overview of the entire project, with integrated querying, filtering and navigation capabilities. The usefulness of visualisation, overview and browsing was demonstrated in that most of the users preferred to simply select concepts in the tree list (or in fewer cases, the hypertext panel) and play the highlighted intervals, despite the simplicity and accessibility of the search field.

However, the graphic visualisation is resource-intensive. Care had to be taken during programming to make the painting of the interval display quick and responsive, particularly when filters were employed. For projects an order of

magnitude larger, this is expected to become a significant problem. The same goes for the computation of derived metadata; it may not be practical to do it in such an automatic and "careless" manner if the project is much larger.

The organisation of media resources could perhaps also be improved; a flat list greater than 20-30 begins to be unmanageable. One solution might be to arrange media resources as individuals of user-specified subclasses of "Media Resource"; this would be a transparent, optional and backwards-compatible change to the current scheme. Another option is to present media resources in a table instead of a list, with their properties as columns, and allow the table to be sorted on the various columns. The problem with this scheme is that it would require more screen space, and that the media resources might not be homogeneously described with properties.

One somewhat surprising comment from the student users was that the ontology used in this project was too big and overwhelming – despite the fact that the hierarchical visualisation of OntoLog/Savanta was designed to allow you to ignore detail you deem irrelevant. There could be many reasons for this – that the ontology *is* in fact too big; that humans have an irrepressible desire to explore beyond what is useful or relevant; that the users do not trust Savanta to aggregate the hidden details correctly; or that this aggregation/hiding is not properly presented and explained in the user interface – so it ought to be studied further.

### 7.1.3 Analysis of television advertisements

In autumn 2003, we tested OntoLog as a tool for analysis of television advertisements in cooperation with the Department of Art and Media Studies at the Norwegian University of Science and Technology. A full account of the project is given in [Engum 2003].

#### Context and purpose

A branch of media studies is to describe and explain film phenomena based on features of the film's production and presentation – lighting, camerawork, editing, composition, *mise-en-scene* – various aspects of the "art and science of motion picture photography", i.e. cinematography [Merriam-Webster Incorporated 2004]. This entails creating painstakingly precise and detailed descriptions of what goes on in the film at every moment – often with second or even frame precision. The aspects that are described are in many cases orthogonal; lighting is independent of camera movement, which is again independent of aural effects such as speech and music.

This description, or logging, is typically done manually, with textual annotations – professor Skretting at the Department of Art and Media Sciences even uses the phrase "stopwatch, pencil and paper" to describe the logging process. In [Thibault 2000], a method is presented where film descriptions are organised in a table, with one row per second, and five columns describing different aspects of each second of film with a semi-structured combination of codes, keywords, symbols and textual descriptions. This comprises an annotation scheme with fixed segmentation, no flexibility, and named descriptors for semantic expressiveness

(see section 3.1), which is not very impressive. Most of the descriptions will last for several seconds, and will in many cases start and end independently of each other. Furthermore, the semi-structured description paradigm is not very suitable for computer-assisted analysis or visualisation. Finally, many of the possible descriptor values are organised in hierarchies, e.g. the camera positions (or movements) shown in Figure 7.4.

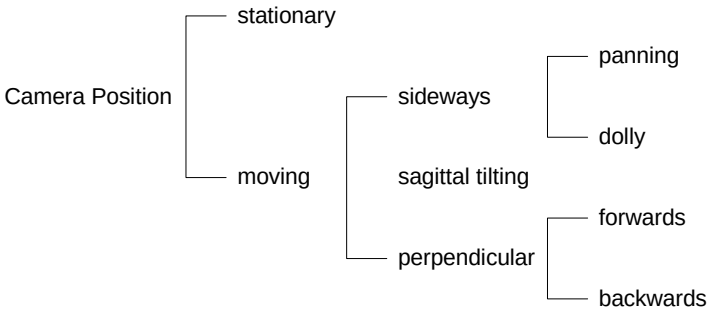


**Figure 7.4: Camera positions (adapted from [Thibault 2000])**

All things considered, we deemed it likely that this form of video content description could benefit from the OntoLog approach. OntoLog's stratified model facilitates the description of several independent features of the film, and its organisation of strata into ontologies meshes well with the way such features are used in film analysis, which will become more evident in the Ontologies subsection.

### Media material

The media material used in this project was four Norwegian TV commercials, one from each of the last four decades. Part of the purpose was to determine if and how the cinematography of such commercials have changed through the years. Apart from their age, the films were intentionally quite homogeneous: they were all around half a minute long, promoted an actual commodity (as opposed to a service, institution, message or similar), and were "photographic" in the sense that they all depicted real people in realistic environments (as opposed to animation, puppets etc.).

### Ontologies

An ontology consisting of 69 concepts was created, most of which is shown in Figure 7.5. It is based on the common terminology for describing film, so it is quite generic – it doesn't contain concepts that are specific to the domain of TV commercials, or to the four films that were analysed. Thus, it is suitable for reuse in other, similar projects.

The four top-level classes are Joins, Sounds, Cinematography (here defined as camera use) and Mise en Scene.

- **Joins** has a set of individuals representing different kinds of editing techniques for joining shots: cut, fade, wipe and others. "Fade in" and

"fade out" could have been placed in their own subclass, "Fade", but the analysers chose not to, for reasons of simplicity.

- **Sounds** have subclasses for background sound, music and speech. Further specification is of course conceivable, but the analysers chose to focus on camera work and editing in this project. Adding subclasses or individuals at a later date will of course not render the annotations in this project invalid or useless, just less specific than they might be.

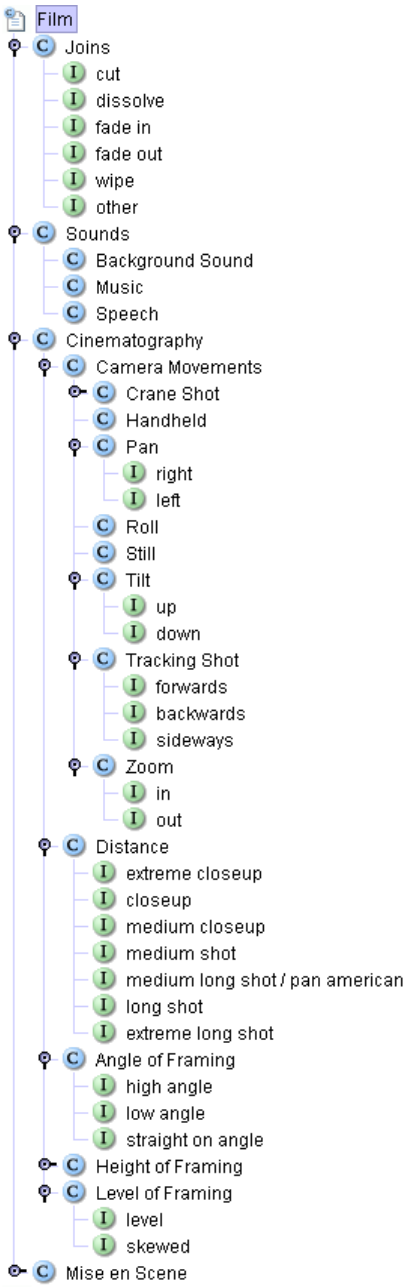


**Figure 7.5: Ontology for film analysis [Engum 2003]**

- **Cinematography** is the largest class, encompassing 40 of the 69 concepts. It describes the properties of the camera in great detail: its movement, distance from the action, angle, height and level.
- **Mise en Scene** (the stage setting – lighting, actors and so on) has 17 subconcepts, but they were not used in this project, so they are not discussed further.

The difference between classes and individuals is not very significant; in most cases, the analysers simply chose to use individuals as leaf nodes in the ontology tree. It could be argued that the resulting semantics are somewhat dubious – is "forward tracking shot" really an instance of "Tracking shot"? Or is it a subclass? What *is* (an instance of) a tracking shot? It is perhaps most correct to say that it is an actual shot in a particular film, but in that case, it is not very useful to include it in the ontology. Perhaps some or most of the individuals in this ontology should be classes (and an *Interval* the instance of a tracking shot), but OntoLog does support changing the "type" of a concept – and this did in any case not affect the work in this project.

**Production**

Almost all of the information in this project is temporal, i.e. intervals and their relationship to the concepts in the ontology. The concepts as such are not described in detail; many of the high-level concepts have comments, but no other properties. Consequently, no properties are defined in the ontology either. Likewise, the media resources are not described with more than a label. A date property (e.g. the Dublin Core *date*, included by default in OntoLog projects) would have been appropriate, since part of the purpose of this project was to compare commercials from four different decades, but presumably the analysts decided to just keep that knowledge in their heads.

The number of intervals described for each film ranges from 21 to 87. The number of intervals overlapping at any given time is usually around three, but sometimes as high as seven. The intervals are typically not described with anything (apart from the obligatory concept), but one interval in each film is labelled with the word "Product", signifying the first appearance of the product being advertised.

The films were logged completely manually, and this was a slow process. The films are short, but must be described in great detail – the "busiest" film has 87 intervals in 42 seconds, an average of over four interval endpoints per second. This led to a lot of pausing and fiddling with the video to place the endpoints accurately, perhaps especially for the "cut" concept – cuts are instantaneous, and were logged as intervals that were as short as possible. On average, more than one minute of logging time was spent for each second of video, which is somewhat disheartening. This might improve with familiarity with the user interface, though.

**Use**

The films (or rather, their annotations) were analysed and compared in the OntoLog tool. Savanta might have been a better alternative, but it was not yet functional at the time. The analysis was performed through visual inspection of the annotation patterns, as well as study of the simple statistics generated by OntoLog, as shown in Figure 7.6. A few of the somewhat interesting findings were:

- The number of joins (and therefore shots) per second is almost the same in three of the four films. This may be a bit surprising, given that short shots and "busy editing" is often considered a fairly modern style. The odd film out is actually the most recent, and consists of a just single shot.
- Two of the films are very different in character – one is staccato, abrupt and energetic, while the other is soft, smooth and flowing. However, the number of shots, the camera movements and even the pattern of the cinematography are strikingly similar.
- The oldest film is made with no camera movement at all, only with differences in distance. So is the most recent, while the two from the eighties and nineties have very intricate camerawork.
- A common pattern for the films is that they start with a long shot to establish the environment, and identify the product with a close-up near the middle of the film.

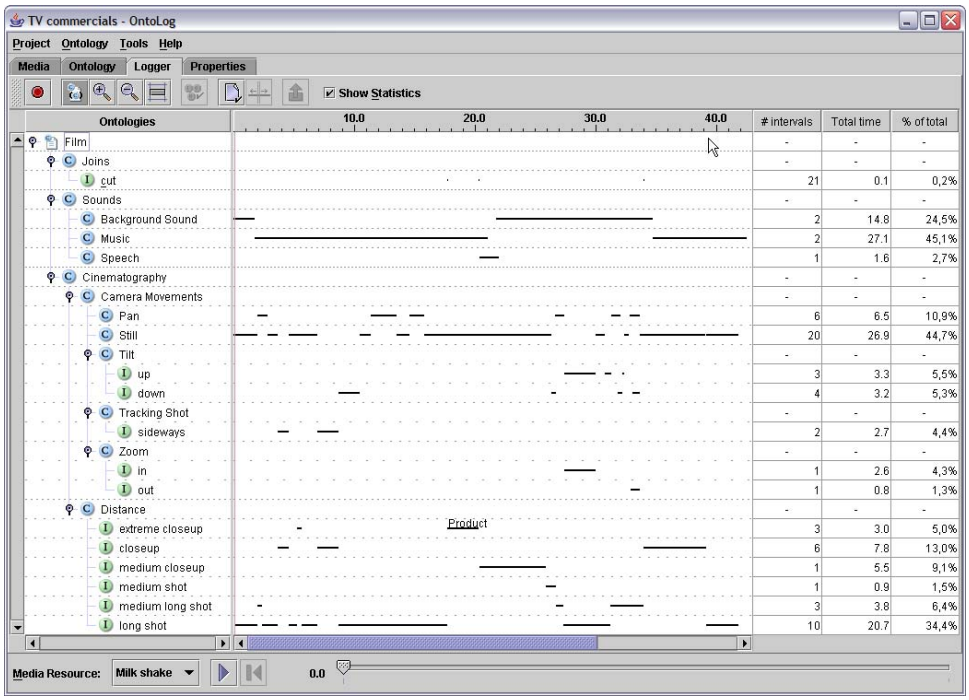


**Figure 7.6: OntoLog annotations of Norwegian milk shake commercial from 1987, showing statistics**

The validity of these observations is of course questionable, given the meagre size of the source material. But that is not the point; the point is that OntoLog facilitates such observations.

**Discussion**

OntoLog was well received by the film analysts; it was described as an "exciting possibility for mapping and displaying overviews of formal aspects of film … [in order to] compare the visual language of different directors, different epochs or different national film cultures" [Engum 2003] (my translation). Unfortunately, the benefits of the OntoLog model and presentation system were somewhat overshadowed by two things: The lack of automatic annotation, and the awkward handling of cuts.

As previously mentioned, creating the annotations was very time-consuming. While not more so than a pen-and-pencil manual annotation, this was problematic in that OntoLog thus was seen as a technologically-intensive solution which did not solve the apparently biggest problem with film analysis: The drudgery of manual logging at this level of detail. Mechanisms for shot detection and/or camera-movement classification would presumably improve the value of OntoLog a great deal for this purpose. See section 6.3 for more on this.

Speaking of shot detection, the other problem was the handling of cuts, or abrupt, instantaneous joins between shots. These were recorded in OntoLog as very short intervals, with the result that they hardly were visible in the Logger interface. In Figure 7.6, for instance, only three of the ten cuts are (barely) visible, due to their tiny temporal extent. A simple "fix" could be to ensure that an interval is never displayed as less than one pixel long, but that is hardly good enough. A cut is a very powerful effect, and is important to establish the "visual rhythm" of a film, so it should have a strong visual presence in OntoLog as well. A suggestion was to extend the OntoLog model to include an "Instant" class – like an Interval, but with just a single time value – and to display Instants as vertical lines in the Logger interface, constituting a grid-like partitioning of the film.

### 7.1.4 Other cases

A few other cases were tried, with less conclusive and complete results.

**Ethnographic analysis of airplane pilot interaction**

OntoLog was used somewhat in a project to study the interaction between airplane pilots – patterns of communication, physical actions, delegation of piloting tasks and such things. There were plans to use it extensively, but the user decided to use the "grounded theory" [Stern 1995] ethnography methodology, and felt that OntoLog did not match this technique very well. Specifically, he was reluctant to define categories (concepts) ahead of time, since this might in some sense "pollute" the analysis of the data by projecting the analyst's expectations and predilections onto the material. Instead, he apparently favoured a less structured approach, letting the categories emerge while repeatedly viewing the films. It is unclear why the possibilities for incremental ontology design were

found wanting, but unfortunately the communication with this particular test user was unreliable and sporadic.

In any case, the user annotated three videos once he had analysed them manually and designed the categories. He remarked that this was a smooth process, and that OntoLog would be very useful in projects where the categories were defined beforehand, but did not use these annotations for anything worth mentioning.

**Study of mid-century infomercials**

In Norway in the fifties and sixties, a number of hour-long infomercials called "housewife films" ("Husmorfilm") were made. These were a mix of advertisements, tutorials and dramatised short stories around the themes of cooking, cleaning and so on. They were shown at cinemas, featured well-known actors, and were very popular. Scientists at Department of Art and Media Studies at the Norwegian University of Science and Technology are studying these films – their use of cinematography, modes of exposition and use of voice-over versus on-screen actors, among other things.

OntoLog is at the time of writing being used for this analysis, but unfortunately the project has not yet matured enough to provide firm feedback on the performance of the OntoLog system for this purpose. Two ontologies of about 40 concepts have been created, and two films have been annotated with respectively 70 and 100 intervals, but the annotations have not been used for much yet. However, some preliminary impressions have been noted:

- The system is perhaps a bit complex. The OntoLog application is somewhat difficult to use; it is not entirely clear how the four main panels interact, and what information goes where. There are some user interface nits to pick.
- The technique of using the same ontology to describe several films can feel strange at first – what if the films use different actors, techniques and themes? – but enables very useful comparisons, once you get used to the notion.
- The user interface is very flexible and robust; being able to edit the ontology while logging is useful, since you thus can create it incrementally and easily change it as your requirements emerge and solidify.

### 7.1.5 Summary

The reason for using temporal annotations was the same in almost all these cases: to perform an analysis of the media material. Only the lecture case had indexing and retrieval as its primary goal. More cases of this kind would increase the validity of the evaluation, but the preponderance of analysis cases might indicate that that is in fact a more common purpose for detailed temporal annotations. That said, it should be noted that all the test cases used the annotations as an index for navigating through the video, and performed the analysis mainly in a visual manner (i.e. not using the analysis functions of Ana, Savanta and OntoLog very

much), which justifies the effort spent on designing and implementing visualisation, browsing and navigation functionality.

The media used in the various cases were very diverse; from TV advertisements of around thirty seconds, through user-produced unedited video recordings with a length of a few minutes to half an hour, to hour-long cinema productions. OntoLog handles them all well, though the short videos expose a few problems with the logging interface, discussed later.

The size and complexity of the ontologies were all about the same order of magnitude – around fifty concepts. The "IT Introduction" ontology was bigger, but should arguably have been split up into several smaller and more coherent ontologies, or even simplified – users indicated that it was a bit too complex. The cinematography ontology was also quite big, but a significant part of this was not used. This might give an indication on how many concepts users find it comfortable to work with, but whether this is given by the design of the user interface, or by human psychology (or both) is an open question.

The time spent on manual annotation of the media varied greatly, from one and a half to sixty times the playback time. The excessive amount of time used in the TV advertisement case was due to the need for very short and very accurate annotation intervals. The video controls in the OntoLog application do not support variable playback rate and frame-by-frame stepping; if they did, the time and effort would most likely have been reduced significantly. Support for instants (see section 0) would also have facilitated production, editing and analysis of the annotations.

As mentioned earlier, in most cases the annotations were used for browsing, navigation and visual analysis; for the most part in the OntoLog application but also (in the "IT Introduction" case) using Savanta. Despite the focus on analysis, and the presence of fairly powerful tools such as Ana, the test users felt that OntoLog alone was sufficient for their purposes. One expressed interest in the possibility for exporting data for use in a statistical package, but was not concerned enough to follow this up with more specific requirements.

In general, the test users were quite happy with OntoLog (except for the lack of fine-grained video control). The interval visualisation, logging interface and ontology editing interface were all lauded, though they all underwent some modifications as a response to feedback from the users. In contrast, the conceptual model was not really changed since its first inception; the users found its expressiveness, preciseness and flexibility good enough to not really have anything to say about it.

## 7.2 Evaluation of Savanta

The design of applications, and user interfaces in particular, is not an exact science where quality can be objectively defined. The subjective evaluation by a group of prospective users is therefore one of the best ways to evaluate the usability of an interface.

This section presents a usability evaluation carried out using Savanta and two other comparable interfaces, *Savantoogle* and *Forms*. The first part presents the setting – the overall purposes of the evaluation, the interfaces to be tested and the evaluation design. The results from the evaluation are presented in the second part, while the third and final part discusses the results and draws conclusions about the usability of Savanta and thus the validity of our approach.

*Note:* The "we" in this section refers to Jon Olav Hauglid as well as the present author; see appendix E.

### 7.2.1 Setting

The purpose of this evaluation is to seek answers to the following questions:

- Is integration of several information gathering methods a good idea? Does it provide significant benefits compared to interfaces that specialise in a single method?
- What is the relative importance of simplicity versus power? Can a powerful but complex system outperform a simpler one with regard to user satisfaction, without significant amounts of user training?
- How does the nature of information gathering tasks affect the suitability and efficacy of such interfaces? What, if anything, changes depending on whether the tasks are simple or complex, specific or open-ended?

As the purpose of the evaluation is to find the strengths and weaknesses of Savanta, a comparative evaluation with other interfaces for accessing temporal annotation databases was a natural choice. By comparing Savanta with other interfaces, one can gain a clearer view of the merits and the weaknesses of the design. Such evaluations also help to improve the reliability of the study by limiting the impact of the Hawthorne effect [Mayo 1933]. This effect states that simply showing concern for users' situation improves their performance. Comparative evaluations cancel out this effect as they measure the relative merits of the interfaces rather than the satisfaction level with a single interface.

We chose to compare Savanta with applications based on two common paradigms for database search:

- Information retrieval as used by web search engines, where search terms entered into a single text field produce a ranked list of documents. This paradigm is represented by Savantoogle, described in section 6.8.
- Construction of boolean query expressions using forms, with an unranked list of matching intervals as the result. This paradigm is represented by Forms, described in section 6.9.

We chose to create our own implementations instead of using existing systems. That way, we would be able to query the same database with all three interfaces, and they would be quite similar in look, feel and polish. This would help the test subjects focus solely on the paradigm differences, not on database differences or implementation details.

### Evaluation methods

In general, the purpose of usability evaluations is to examine the usability of one or more user interfaces. As a wide variety of different evaluation methods exists, a fundamental task in evaluation design is to decide which methods to use. Broadly speaking, existing methods can be divided into two groups based on which kind of information the method provides. *Qualitative methods* such as interview and observation are concerned with understanding how an interface works – how users perceive the presented information, why users sometimes get stuck, etc. *Quantitative methods*, on the other hand, are used to *measure* quality – how many users prefer a given interface over other interfaces, how fast a given task can be solved, etc. Examples of quantitative evaluation methods include performance measurement and questionnaires. For an in-depth description of available evaluation methods, see e.g. [Preece et al. 1994].

Based on the purpose of this evaluation, as defined in section 7.2.1, this evaluation was divided into two parts: a pilot study and a main study – each using different evaluation methods.

*Pilot study*

Before the actual evaluation phase started, a pilot study was performed. This included presenting the three implemented interfaces to three fellow researchers and having them comment on any potential usability problems they could spot. This also included an evaluation of the main study design. In this way, the pilot study can very well be viewed as a part of the implementation phase rather than the evaluation phase.

As none of the interfaces had been used by anyone but the designers, the pilot study was a critical phase in assuring a consistent level of quality among the implementations. Several iterations of implementation and inspection were necessary before the implemented interfaces were ready for the main study. In particular, several problems regarding consistent wording of labels and buttons as well as confusing general interface layout, were fixed.

*Main study*

As is already evident, the main study is a comparative evaluation of three different interfaces for information access in temporal annotation databases. For comparative evaluations, one has the choice of having each test subject evaluating a single interface (between-subjects) or have all testers evaluate all interfaces (within-subjects) [Mitchell and Jolley 2001]. For this evaluation, the latter alternative was chosen – primarily to reduce the number of test subjects needed and to assure that individual differences are cancelled out (e.g. some test subject being more positive in general than others).

Further, to remove ordering effects, a counterbalanced design was used by which the test subjects were randomly divided into three groups. All members of each group tested the interfaces in the same order, and this order was altered between groups such that all interfaces were equally often tested first, second or third. Each of the three test groups consisted of three test subjects – nine in total. This

number was kept fairly low as the evaluation was primarily qualitative and thus required less testers and more time per tester. As the database used for all interfaces contained video and annotations from a computer science class, all test subjects were students that had recently taken this class. They were thus familiar with the subject at hand and would be legitimate end-users of the implemented interfaces.

The main study itself consisted of three steps. First, the test subject was given a brief introduction and tutorial for one of the interfaces. Second, the subject carried out several predetermined tasks using the interface while being observed. These two steps were then repeated for the two remaining interfaces. The training of the test subjects was done to equalise the playing field and to lessen the impact of individual differences with regards to previous experiences with similar interfaces. Finally, a questionnaire concerning all three interfaces was handed out. Thus, the main study made use of two evaluation methods: Observation and questionnaires.

The objective of the observation was to gather qualitative data about the interfaces. This was done by recording observational data on paper while the evaluation tasks were performed by the test subjects. These notes were often supplemented by a short informal interview afterwards to clarify some of the observations made.

The questionnaire was used to get quantitative data concerning the relative performance of the three interfaces. For this reason, all questions were repeated for the three interfaces and the test subjects were instructed to focus on comparative evaluation (i.e. which interface was best in a given category). All but one of the questions were taken from the comprehensive Questionnaire for User Interaction Satisfaction (QUIS) [Chin et al. 1988].

### Evaluation tasks

One of the stated purposes of this evaluation was to study the performance of the interface with respect to different types of information gathering tasks. Three different categories of tasks were defined: Simple retrieval, complex retrieval and exploration. These are based on and correspond roughly to the task types defined in [Shneiderman 1997]: Specific fact finding, extended fact finding, and open-ended browsing / exploration of availability.

We defined *simple retrieval* as tasks where the user is looking for a simple answer – yes or no, or a single interval of video containing something of interest – and where there is no need to combine search terms or consider relations between things. Examples include finding the sole place where a certain term is used, or determining if a given term is mentioned in a particular set of videos. Tasks of this kind are conceptually simple, and a good query interface should be able to handle them simply and efficiently. If an interface is geared towards more complex queries, it may inhibit the formulation of simple ones – it may be too powerful for its own good. Therefore, we expect the simplest interfaces – Savantoogle, and to a lesser extent Forms – to score well for simple retrieval.

*Complex retrieval* is here defined as tasks where the result is more intricate – a set of video intervals, for instance, or an aggregation of them (say, total length) – or where the user needs to combine search terms, or establish how different terms relate to each other. Determining how much time is spent on a given term, or finding intervals where two particular terms are active at the same time, are examples of complex retrieval. This calls for interfaces able to construct composite query expressions and to perform aggregate functions. We expect Savantoogle to perform worse than the other two interfaces at this, since it has little support for aggregation and temporal operations – it considers video documents the unit of retrieval. It is more difficult to predict how Forms and Savanta will perform.

*Exploration* is tasks where neither the goal nor the path towards it is entirely clear. It may be the user trying to find out what exists in the database, what trends can be established, what characterises a subset of the database. Examples include finding out what the most important term is in a given set of videos, or determining which narrower term of a particular term is most used. For this kind of fuzzy fact-finding, we expect the visualisation and multiple access methods of Savanta to give it the upper hand.

### 7.2.2 Results

This section presents the observations made during the evaluation as well as the questionnaire results. Finally, a discussion of the outcome of the evaluation and a look at reliability and validity concerns.

#### Observation

By observing the test subjects, listening to their comments and interviewing them afterwards, we discovered a few interesting tendencies and patterns. Many comments were related to details of each interface, such as button placement and double-click behaviour. This is not in itself very interesting outside the context of our implementations, but in many cases, more general guidelines and tendencies may be distilled from them. Other observations are inherently more general, and thus more directly useful. In the following discussion, we have tried to abstract the observations away from implementation details, and to focus on the more high-level lessons to be learned from them.

*Applying conventions makes sense* – the users were pleased with finding elements of software systems they had used before. The Home, Back and Forward buttons of Savanta, familiar from web browsers, were used extensively and without any problems. This was expected, but it is nice to see such expectations confirmed, especially since these controls in this case were used outside their normal environment, web browsers. Savantoogle was praised for its resemblance to Google, which everybody was familiar with and enjoyed using. Likewise, most of the test subjects expected something useful to happen (video playback, mostly) when they double-clicked on search results and other things; it was a source of annoyance and confusion when it didn't.

*The model and/or domain have clear affordances* – given the organisation of the information in our database, our test subjects expected the user interfaces to be organised correspondingly. The temporal aspect and the list of videos were not directly accessible in Forms and Savantoogle, and this was frowned upon. For instance, to find a particular video in Forms, the users had to perform a query using a disjunction of all the top-level terms, and scan the result manually to find the desired video. This was often referred to as "cheating" and a roundabout, unintuitive process, though it was not particularly difficult or labour-intensive to do. The lack of a visual, temporal overview of the media resources in Forms was also lamented.

*Feedback is important* – perhaps not a stunning discovery, but the users clearly had a strong need to be certain that their queries produced correct results; to know why the interfaces came up with the results they did. When using the search field in Savanta, some test subjects spent considerable time expanding the term tree to ascertain that it actually was a matching term that had been found. When querying in Forms, a few test subjects wanted to see which terms overlapped the search result, for instance with an interval visualisation like in the other two interfaces. In Savantoogle, many repeated essentially the same query several times, varying the use of quote marks and pluses, to be sure that the desired result had been produced. This could perhaps be improved by presenting a better summary of the media resource for each match – our implementation just presented its title and comment, and the list of matching terms.

*Multiple access methods is good* – many felt straight-jacketed by Forms and Savantoogle, since those systems only had one method each for finding information. Several users commented that it was easy to "get stuck", whereas in Savanta, the users had multiple angles from which to approach the problem. While this meant extra complexity – some commented that it was difficult to remember all the possibilities – it was clearly preferred, and "felt faster" (even when it was not). This confirmed our hopes that a richer, more powerful and flexible environment could still be at least as pleasant to use as a simple, minimalist one, even for simple tasks.

*Complex tasks and simple interfaces don't match* – when faced with a complex or exploratory task, many of the test subjects wanted to solve it in a correspondingly clever manner. They did not like to use a sequence of simple operations, especially when they had to assemble the result manually in their heads (or on paper). Even though it might be easy and obvious, it felt like drudgery. In many cases it was, of course – finding the most used term in a subset of the database is very time-consuming when the interface only supports elementary functions like searching for a word. It seems that users expect and require the system to provide functions at the same level of complexity as the tasks the users want to perform, even though they have to spend time finding them and learning to use them.

### Questionnaire

The questionnaire consisted of 10 questions drawn from QUIS [Chin et al. 1988] for each of the interfaces, 30 in total. The following questions were used:

1. Reactions to the system with respect to simple retrieval (terrible – wonderful).
2. Reactions to the system with respect to complex retrieval (terrible – wonderful).
3. Reactions to the system with respect to exploration (terrible – wonderful).
4. Overall reactions to the system (terrible – wonderful).
5. Learning to operate the system (difficult – easy).
6. The expressiveness of the system (inadequate – adequate).[1]
7. Amount of information that can be displayed on screen (inadequate – adequate).
8. Messages which appear on screen (confusing – clear).
9. Tasks can be performed in a straight-forward manner (never – always).
10. Amount of help given (inadequate – adequate).

The results of the questionnaires are displayed in Figure 7.7.

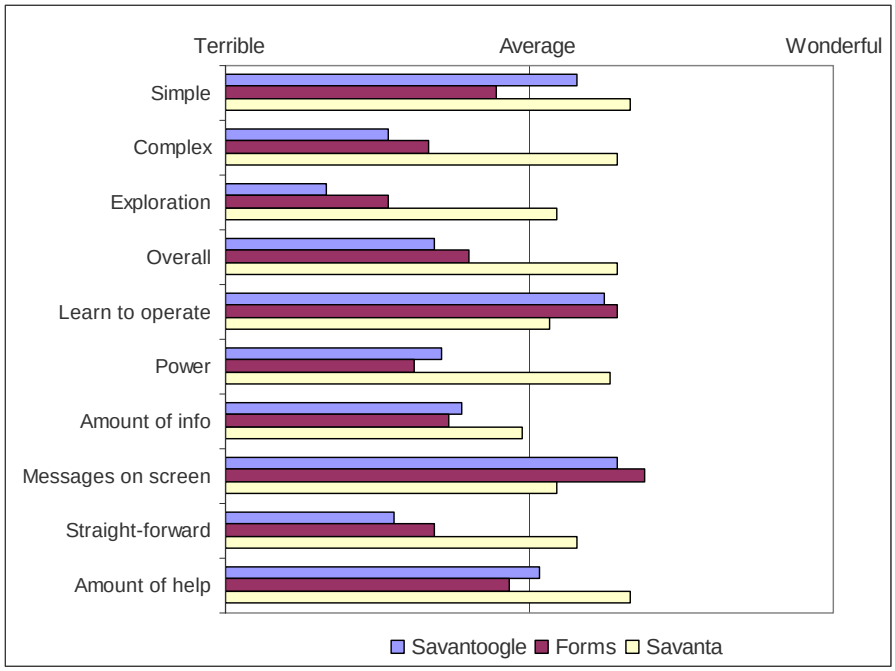


**Figure 7.7: Results from questionnaires.**

Reviewing these results, we have found the following three characteristics to be most evident:

[1] This question was not drawn from QUIS

- **Ranking with respect to task types**
  Overall the picture is reasonable clear: Savanta was best liked regardless of type of task. The other two interfaces were found to perform almost equally.
- **Savanta best for more complex tasks**
  The largest differences were evident for complex retrieval and exploration tasks, while Savanta and Savantoogle performed almost equally well for simple retrieval. Not surprisingly, there is a clear correlation between the results from the "Complex" and "Power" questions.
- **Learn to operate vs. Straight-forward**
  On first glance, there seems to be an inconsistency between Savanta's results for these two categories. How can both be the most difficult interface to learn and the most straight-forward to use? We believe the answer lies in the results from the "Power" question. While a simple and less powerful interface might be easy to learn, it can require a huge number of actions in order to complete complex tasks. With Savanta, once the user had grasped the interface, complex tasks could often be completed in just a few steps.
- **High power wins over high complexity**
  As is evident from the "Learn to operate" and "Messages on screen" questions, the test subjects found Savanta more complex and difficult to use. The power of the interface and the straight-forward way even complex tasks could be performed did however make Savanta the preferred interface regardless.

### 7.2.3 Discussion

Integration of several access methods is the cornerstone of the Savanta design. The results of the usability evaluation strongly suggest that this was a good idea. Offering multiple ways to reach one's goal made it less likely for users to "get stuck" using Savanta compared with the other two interfaces. Further, the results indicate that graphical visualisation of temporal metadata greatly helps the users in quickly gaining an overview of complex data.

While these findings were expected, it was interesting to examine the usability consequences of having multiple methods for accessing data. Savanta's interface was found to be a bit more complex, but the users found the increase in expressive power more than made up for this.

Even as Savanta in general was the best liked interface among those tested, there were variations depending on information gathering task. For complex tasks as well as exploration, the multitude of possibilities of Savanta ensured that it was the clear winner. But for more simple tasks, more simple interfaces performed equally well.

### Reliability and validity

The results of a usability study are not worth much if the evaluation design is flawed. In general, possible methodology pitfalls can be classified as either reliability or validity concerns [Nielsen 1993]. An outcome is *reliable* if a new evaluation gives the same result, while *validity* is defined as whether the result actually reflects what one really wants to test.

The most obvious way to improve the reliability of our evaluation would have been to use more test subjects. People are different, and the impact of these differences is reduced when more testers are involved. Still, the use of a comparative evaluation design reduces the impact of individual differences.

As for validity, the type of test subjects used is a possible source for concern. All test subjects were quite experienced with computers and thus not the best to judge usability with respect to other types of users. However, with the test data we used, the test subject very much represented the intended target audience.

Another possible concern with respect to validity is the choice of the other two interfaces used for comparison. While their design and capabilities were tailored to match existing classes of interfaces for information access in temporal metadata databases, they were still designed and implemented by us. Thus (unintentional) bias on our part is a clear possibility. Still, we felt it was critical for comparison purposes that all interfaces used the same data (and conceptual model) and making our own implementations was pretty much the only way to achieve this.

# 8 Discussion

In this chapter, the various findings of this work are discussed. The first section is structured according to the research questions put forward at the beginning of this thesis, while the second part discusses the research approach itself, as well as experiences concerning the use of RDF for video annotation.

## 8.1 Research questions

The research questions posed in section 1.2 are here reviewed, along with discussion on how (or if) and how well they have been fulfilled.

### 8.1.1 Video information modelling

The questions put forward were: *How should a model for semantic, temporal annotations be constructed, given that it should be usable for quite different domains, purposes and levels of detail?* What are the requirements concerning such a model? What kinds of semantics, what level of expressiveness should be supported? How can the model be made flexible or extensible enough to support different kinds of applications, while not sacrificing simplicity and usability?

The answer suggested by this thesis is in short: *Use ontologies* – relatively small, custom-built, user-specified ontologies. This enables the users to construct their own universe of discourse, with exactly the desired perspective, level of detail and expressiveness. The coupling of this with *unconstrained temporal intervals, in a stratification based on the structure of the ontologies*, gives rise to a very flexible model. By using the subclass-of and instance-of relations of ontology, annotations with appropriate specificity can be created, and the same structure can be used with significant benefit for visualisation (including zooming and overviews) as well as reasoning during retrieval and analysis.

The model can be thought of as a framework or meta-model, capable of implementing most, if not all, aspects of the existing models described in chapter 3. Its suggested handling of spatial annotations (described in section 6.4) is perhaps not as elegant and powerful as that of BilVideo, but that may be the price to pay for flexibility and user-friendliness in other areas (or possibly a reflection of the relatively little importance assigned to the study of spatial annotations in this thesis). The model has no particular mechanism for hierarchical segmentation, which is a common feature in several older models, but that can be implemented using appropriate classes representing the hierarchy levels of shot, scene and sequence, if desired.

Simplicity and usability was also a goal. This, I think, has been reasonably successful; the model does not have very many fundamental entities and relationships, and most of them are easy to grasp. The difficulty lies in the use of ontologies, which may be unfamiliar ground for many video annotators, and the meta-model aspect of it: That before you can start describing the video you have to create something to describe it *with*. In practice, however, this has not been a problem; users are only too happy to be able to organise their knowledge in their own way. The possible confusion caused by the distinction between subclass-of

and instance-of relations can if necessary be glossed over; indeed the whole concept of ontology can be "dumbed down" to a simple "hierarchy of keywords", as is done in Savanta. The process of defining (and, to some extent, using) custom properties is a bit difficult, perhaps because of too little effort in designing the user interface for it, but in practice, few properties beyond "label", "comment" and the Dublin Core properties are used or desired.

### 8.1.2 Tools and interfaces

The questions put forward were: *How can one create an infrastructure to handle the various tasks related to the usage of semantic content annotations?* What kinds of architectures and technologies can or should be used? How should the metadata be stored, accessed and transferred to the user? How should tools for entering, browsing, presenting and querying annotations be constructed?

The answer suggested by this thesis is not as easy to state succinctly as the answer to the previous set of questions. Some of the infrastructure is dictated by the model – tools for defining projects and constructing ontologies must be provided – and this thesis cannot claim to contribute very many original thoughts in that area. However, the technique for visualising annotations based on using the hierarchical ontology terms as strata, is novel and very useful. It enables a highly needed overview and zooming functionality for temporal annotations, and is intuitive and natural to use, both for annotation production, browsing and analysis.

Information gathering tools have also been studied to a respectable extent. Various techniques for search, browsing and analysis has been investigated, culminating in Savanta, where modern user interface paradigms and interaction mechanisms are integrated to form a novel, rich environment for information gathering in temporal annotations. Savanta may be criticised for its complexity, but simpler tools are available, if they are more appropriate: The thesis has shown that the proposed model supports quite diverse retrieval tools, with different strengths and weaknesses.

As for architecture, storage and transfer issues, the thesis suggests and proves the viability of one option: RDF(S) stored in relational databases, and managed by Jena. RDF is an excellent match for the OntoLog video annotation approach, since it provides both a simple, unified information representation scheme, rich possibilities for incremental, iterative model customisation and refining, and a well-defined language for formal specification of conceptual models and ontologies. Despite the immatureness of RDFS and the tools supporting it, this has been a success. Further development of RDF, RDFS and related standards, and of tools supporting them, will likely improve this even more.

### 8.1.3 Usability and performance

The questions put forward were: *How does such a system fare in practice, in the real world?* Does it provide real benefits compared to prior approaches found in literature? Under what circumstances is it appropriate and successful, and when is it not? Is there an actual need for this kind of system? Are the model and the tools

user-friendly enough? Expressive enough? Extensible enough? Simple enough? Is the model possible to index and search efficiently? Is it scalable? How does the choice of architecture/technology affect this?

The answers to this set of questions are perhaps not as universally interesting as the previous two, but they may be useful to consider nonetheless.

*What are the benefits?* – The benefits the model provides are mainly flexibility and expressiveness. The users may create their own descriptors, rather than use predefined, broad categories such as "event" or "object". Compared to other models that also allow such customisation, the OntoLog model scores points for being based on a formal theory for describing (aspects of) the world – ontology – which in theory makes it possible to reason about the descriptors and, thus, the media they describe. It is also very open, in the sense that there are no limits on how complex or specific the ontologies may be, apart from the constraints dictated by the ontology language used. On the other hand, for some kinds of data and purposes, the model is not very suitable, as mentioned in section 8.1.1 and discussed further below. The basic model is reasonably simple, but that may not be very important; the tools and user interfaces wrapping a model is at least as important to the usability of the system as a whole, as the intrinsic complexity of the model.

As for the tools and interfaces, it is harder to say what benefits they offer compared to other approaches, due to the heterogeneity of this field, and the constraints put upon the tools by their underlying models. Interviews with actual users suggest that OntoLog is more powerful and user-friendly than commercial products with similar goals and functionality (see 7.1.1). It is difficult to make such comparisons fair, especially given OntoLog's goal of being appropriate for very different purposes; however, the technique of hierarchical aggregation of annotations in a timeline view is a valuable contribution, since it enables rich, interactive and zoomable overviews of video annotations, and thus also inter-video browsing. The information gathering interface Savanta has also proven itself superior to more traditional approaches, as discussed in section 7.2.

*Under what circumstances does it work? Is it adequate?* – Despite OntoLog's goal of generality and adaptability, there are some circumstances it does not handle perfectly. Due to its interval-based model and timeline-base visualisation, the intervals should be no shorter than 1/50 to 1/100 of the total length of the media file; otherwise, the intervals become too short to be noticed and manipulated in the timeline view. For the same reason, it is ill suited to annotate instantaneous events in videos, such as cuts and transitions, as discussed in section 7.1.3, but this critique is applicable to most interval-based annotation systems. OntoLog also has no direct support for hierarchical segmentation, which is a common feature in older annotation systems.

OntoLog is designed for overlapping annotation intervals. The use of ontologies as hierarchical strata makes it simple and natural to describe the various (and often independent) aspects of the content of the video, and the visualisation technique displays the result in a helpful manner. However, there should not be too much overlap – more than 6 or 7 "active" intervals at any time will cause

problems with the visualisation in its current implementation, because of space constraints. This could however be remedied by assigning more space to each strata (or perhaps just the "busy" ones), or using some other indication of overlap in addition to line thickness – colour, for instance. In practice, this has not been a problem. On the other hand, too little overlap is not ideal, either – the analysis functions of Savanta are based on computing relative overlap between sets of intervals. If no overlapping annotations are ever needed, it might be simpler and more appropriate to use a model based on user-defined segmentation.

The effort of constructing ontologies is another point to consider. One would not want to construct a separate ontology for each video to be described; this would be too much effort, and severely undermine the usefulness of tools like Savanta. OntoLog is designed to annotate a set of related media resources using the same ontology/ontologies for them all. It lends itself well to projects where the purpose involves some kind of generalisation of the content of a homogeneous corpus of video, perhaps where comparison or analysis of the videos is part of the process. It is not particularly suited for indexing an arbitrary collection of videos with very different contents and properties, though a very big or very general ontology could be used.

Among media researcher, video transcription is a common task. Transcription creates a textual, non-temporal representation of the video content, which is then used for further study. OntoLog provides no direct support for this; however, it should be investigated whether this use of transcription is merely to avoid tedious winding and rewinding of analogue video tapes, possibly rendered obsolete by digital video and indices based on temporal annotation.

*Is it efficient and scalable?* – This has not been studied very well. The OntoLog system works well in practice, at least for projects the size of the lecture recording project discussed in section 7.1.2. The nature of the RDF model, and the techniques used by Jena to store it in a relational database leads to a lot of joins even for the simplest query, but the potential performance hit caused by this is sidestepped by keeping all the data of a project in memory. The algorithms and data structures used in the visualisation have linear complexity and storage requirements, but some of the analysis algorithms in Savanta have quadratic complexity. This could be a problem for bigger databases, but it is hard to avoid if you want that kind of functionality. Savanta, being very visually oriented, is in any case not particularly suited for huge databases.

## 8.2 Other issues

Apart from the answers to the research questions, some other issues are also worth discussing. The research approach had of course great impact on the results presented in this thesis, so an examination of what could have been done differently is presented in 8.2.1. Subsection 8.2.2 reviews how well the scenarios from section 4.1 have been fulfilled. The use of RDF for video annotation is (though not an explicit research goal) an interesting and novel topic; it is discussed in subsection 8.2.3

### 8.2.1 Research approach

The questions asked in section 1.2 are rather general, perhaps even a bit vague. Consequently, the answers I have given in the previous section may seem correspondingly general and vague. Could I have asked different questions, perhaps more specific ones? Or could I have produced more specific results with greater validity by pursuing some other research approach?

I could perhaps have focused on one particular domain/purpose for temporal content annotation – say media research on Norwegian TV commercials and infomercial films. This would have enabled me to study this domain in detail, establish clear requirements, and determine my success and my contributions according to this – more clearly, and perhaps more quantitatively. However, my vision of adaptability and flexibility would most likely have fallen by the wayside with this approach, and I believe that that idea is a worthier goal than creating a better single-purpose tool.

But could I have produced more reliable and valid results? Two studies suggest themselves as possible improvements:

- I could have performed a quantitative evaluation of the main OntoLog application, in order to get a more certain indicator of its quality, preferably compared to other systems.
- I could have investigated the scalability and performance of the OntoLog system more thoroughly, by generating large databases, by computing the complexity of algorithms and access paths, or by simulation.

However, creating a valid and reliable usability test of OntoLog is not an easy matter. Systems for semantic content annotations (and users of them) are rare, so it would be difficult to find an adequate number of appropriate users. The test would probably have to be constrained to one kind of domain and purpose, since the creating of content annotations is a task for domain experts, making it even harder to find enough suitable test subjects. It is hard to perform a comparative evaluation, since few (if any) existing systems have capabilities sufficiently similar to OntoLog. All in all, I felt that a more informal approach toward evaluation, based on observation, expert review and interviews would be more fruitful, despite the lack of hard numbers.

In contrast, scalability and performance would have been easier to measure. That this was omitted is partially a result of priority and time constraints. It is also due to the fact that OntoLog ended up as a system for small to medium-sized video databases, as discussed in section 8.1.3. Its model and tools is best suited to a manageable set (in order of tens or hundreds, not thousands or millions) of fairly homogeneous videos, as are most of the tasks for which detailed semantic annotations are needed. For gigantic, heterogeneous video archives like those of broadcasting companies, OntoLog is not suited. Therefore, I did not consider it essential to analyse how OntoLog would handle huge amounts of data.

The iterative, design-based research approach has functioned well. The system has gone through several iterations (the main OntoLog application went through

six major versions, and countless minor ones) based on informal expert reviews and pilot studies, leading to both a better system as well as a better understanding of its requirements and success criteria. It is hard to imagine how the alternative research approaches defined by [Denning et al. 1989], theory and abstraction, could have been more successful than the hands-on, practical approach in this particular case. The success of OntoLog is determined by the acceptance of actual users of semantic video annotation; to obtain that, an actual system is more effective than a theory or an abstract model.

[National Research Council 1994] defines three primary purposes for building systems such as OntoLog:

- Proof of performance – that an implementation is quantitatively better than other implementations in terms of performance or improvements and enhancements.
- Proof of concept – that a certain (preferably original) idea or approach is viable.
- Proof of existence – to demonstrate a novel computing phenomenon.

The OntoLog system falls in the second category, proof of concept, with perhaps some aspects of proof of performance. This thesis shows that ontology-based stratified annotation and visualisation is a good idea, and that the information gathering approach of Savanta outperforms a few other common approaches under certain circumstances. However, as NRC [ibid.] points out, it is harder to prove the value of proof-of-concept artefacts than proof-of-performance artefacts, since their contribution is often qualitative and based on subjective human opinion. The evaluation of Savanta produces hard numbers, and could be viewed as a proof of performance, but the numbers are based on the judgment of the test subjects, not on objective measurements.

In summary, the research approach used in this thesis has been appropriate and successful, but its results may be considered somewhat "soft", qualitative and vague. One possible improvement might be an abstraction- or theory-based formal analysis of the properties of the OntoLog model, with respect to (say) expressiveness or performance.

### 8.2.2 Scenarios revisited

The development of the OntoLog system has been user-driven and iterative rather than scenario-based. However, as the scenarios from section 4.1 were used to explore the problem space and justify requirements, it may be interesting to review how well each has been fulfilled by the OntoLog model and the current set of tools.

#### Support for movie watching

As described in section 5.2.1, the OntoLog model is clearly expressive enough for Jon's demands. Actors and producers, their names and relationships to films are

easily handled by OntoLog's user-defined classes, individuals and properties. Defining action types as classes and connecting them to intervals is also simple.

However, the current set of tools is somewhat inadequate: Jon requires better query functionality for non-temporal information. The focus in this thesis has been on the temporal information; however, constructing such a non-temporal query interface should pose no big challenge. For intra-video navigation (finding the next fight scene) both the OntoLog application and Savanta are sufficient. A simpler, more specialised interface might be more appropriate, though.

**Managing interview recordings**

Cathrine's indexing and commenting needs are met nicely by the OntoLog system. Her topic hierarchies are easily expressed using OntoLog's ontologies, and plain-text comments can be added to both intervals and concepts without hassle. The OntoLog application provides a suitable environment for the indexing and annotation of her audio files; Savanta may additionally be used for browsing and reviewing if the OntoLog application is found wanting in this regard.

**Support for system analysis**

Ian's requirements are also fulfilled reasonably well; the OntoLog application provides ample means for detailed and incremental creation of an ontology used for indexing video. It is easy to add more classes, reassign intervals to new concepts and restructure the ontology, even during interval creation. Using Savanta, Ian's team can quickly locate all video intervals related to arbitrary combinations of concepts in the entire project, in order to study them.

However, the usability of the OntoLog application with regard to creating and using custom properties and relationships should be studied further. It may also be a problem that OntoLog is a single-user application; it does not support multiple annotators.

**Video-based system evaluation**

The fulfilment of this scenario is described in section 7.1.1. To sum up, it was a success.

**Lecture database**

Likewise, the success of the lecture database scenario is described in section 7.1.2. It should be noted, though, that not all aspects of it were performed in practice: Little extra information were added to the intervals and topics, and the possibility of collaborative annotation (students entering their own notes and comments) was left unexplored.

**Police investigation**

Like Ian, Jostein is served well by the ontology capabilities of the OntoLog system. Section 5.2.6 details how the ontology creation and annotation is performed; the OntoLog application supports this perfectly.

However, the tool support for analysis is found somewhat wanting. Ana and Savanta provide mechanisms for temporal analysis, and Savanta is particularly apt for finding correlations between the appearances of different concepts, but they are worse at comparing concepts based on their properties, as Jostein needs. "Merging" concepts – replacing two concepts with their union with respect to properties and intervals – is a function that is useful for Jostein, but must be performed manually in the OntoLog application.

**Summary**

The needs of the scenarios have in general been successfully fulfilled, but more so for the medium-complexity scenarios than the simplest and the most complex. For the movie watching scenario, the tools provided are too complicated, and a more specialised design should be considered. For Ian and Jostein, there is greater uncertainty regarding if the tools are powerful enough as well as sufficiently usable. However, the OntoLog annotation model is suitable in all the scenarios, and the Cathrine, Hallvard and Steinar scenarios are very well served by the current set of OntoLog tools.

### 8.2.3 Using RDF for video metadata

Using RDF for video metadata has both advantages and disadvantages. It is very flexible and expressive, due to the mechanism RDF has for defining its own classes, instances and properties. Both the OntoLog model itself, the ontologies and the actual annotations are defined in the same language and stored in the same database; a very homogeneous and elegant solution. However, RDF data is somewhat verbose and intricate. To define an interval (and its relation to a concept) in OntoLog, five statements are needed[1]:

| | | |
|---|---|---|
| video.mpg | o:hasInterval | int1 |
| int1 | rdf:type | o:Interval |
| int1 | o:from | 15700 |
| int1 | o:to | 21800 |
| int1 | o:relatesTo | conceptX |

In Jena, this information is stored as five rows in a Statement table, where both the subject, predicate and object reference rows in a Resource table (or a Literal table, in the case of the objects of the o:from and o:to predicates). Reading a single interval from the database thus entails joining the Statement table to itself and the Resource table several times. A more efficient storage scheme would of course be to have a dedicated Interval table, with "from" and "to" as attributes, and foreign keys relating it to the MediaResource and Concept tables. This would preclude the model from being used by third-party APIs and reasoning frameworks such as Jena, but it would make the model less costly to query.

[1] Strictly, just four are needed, since the rdf:type property may be inferred from the domain of the predicates describing the interval, and/or the range of o:hasInterval, cf. the footnote on page 103.

A less destructive simplification is to treat the intervals not as distinct entities in their own right, but rather as fragments of the media resource, putting the boundary points of the intervals in the fragment part of a URI based on the media resource:

| | | |
|---|---|---|
| video.mpg#15700-21800 | rdf:type | o:Interval |
| video.mpg#15700-21800 | o:relatesTo | conceptX |

This would more than halve the number of statements needed to represent an interval (especially if you consider the rdf:type statement superfluous), as both the interval's endpoints and the reference to its media resource would be implicit in its URI. The fragment identifier would need to be parsed, but this is little effort compared to a table join. A bigger problem is that the interval's URI would have to be changed if the user edited its endpoints. URIs aren't supposed to change; Jena, for instance, has no support for this in its API. If the URI is used as a join attribute in the database tables, any changes would have to be cascaded throughout the database. It is also worth mentioning that this way of encoding a temporal interval as a fragment identifier is non-standard; other programs would not know how to interpret it.

Using Jena has been a mixed blessing. It has an adequate Java API, which isolates OntoLog from storage specifics – switching from MySQL to Oracle is a matter of changing the parameters of two method invocations. However, it is still under development, and has exhibited plenty of bugs during the construction of OntoLog. The new version, Jena 2, promises better reasoning, OWL support and a database model with increased performance, which all would benefit OntoLog.

The use of standards such as RDF enables integration and interoperability with other information systems based on the same standards. However, this has not been a big issue with OntoLog. Users familiar with RDF have commented that OntoLog is a nice tool for creating and editing RDFS ontologies, but as the use of RDF for temporal content annotation is a fairly novel approach, other significant synergies have not been forthcoming. Ready-made ontologies are available on the Internet, and can be imported into OntoLog. Users prefer to construct their own, though – custom-made ontologies are smaller and more focused, and using standard ontologies has had no particular benefit in the real-world cases OntoLog has been used. The use of Dublin Core as a default ontology for resource description is nice, but a small thing. However, the possibilities afforded by exporting projects and ontologies as XML and using XSLT to transform them into e.g. web pages are interesting, and might be studied further.

All in all, using RDF for annotating video has been a success. The biggest problem is its performance for huge databases, but this is a problem for RDF databases in general, not just for OntoLog. It is also worth mentioning that the basic idea of OntoLog – using ontologies as hierarchical strata in a stratified annotation scheme – is not dependent on RDF; another ontology definition language might work just as well.

# 9 Conclusion and further work

This chapter sums up the contributions of this work, and suggests possible directions further work might take. The overarching goal of this project, if one looks one level up from the research questions, is to facilitate the use of video in knowledge work, through the use of computer-supported semantic content annotations. It is too early to tell if this goal has been reached, but quite a few more concrete wishes have been fulfilled.

## 9.1 Contributions

The main contributions of this work can be summarised as follows:

*A flexible, expressive, structured and simple model for semantic annotation of temporal media.* Augmenting traditional stratified annotations with ontologies, the model is strictly more expressive and flexible than existing models (ignoring spatial annotations), since new objects, object classes and descriptors can be defined by the users, with formal semantics, on the fly – while not being significantly more complex. Annotation strata represent concepts, classes and objects at potentially many different levels of detail, allowing the video to be described as coarsely or finely as desired – topically as well as temporally.

*Demonstrations of this model's viability in several domains.* The model has proved its worth in several real-world projects from quite different disciplines: medical informatics, education and media research. The ability to let the users define their own classes, objects and properties of interest allows the model to be easily tailored to any domain and purpose.

*A powerful visualisation scheme for stratified temporal annotations in video databases, enabling better overviews and inter-video browsing.* The annotation strata are organised hierarchically in an ontology, with subset (subclass) or member (instance) semantics. This makes it possible to aggregate the visualisation of related strata, making the representation more compact and hiding unnecessary details. The compact representation gives room for visualising and browsing several videos at once, and the users may easily disaggregate or "zoom in on" the strata they are particularly interested in.

*A novel and powerful information gathering interface, based on integrated browsing, navigation, visualisation, searching and filtering, with demonstrated benefits.* Savanta provides a rich environment for video browsing and analysis, utilising the richness of the model, the powerful visualisation scheme and context-sensitive temporal analysis of the annotations. Its flexible and iterative interaction model provides a user-friendly approach to complex information gathering needs.

*A demonstration that RDF is a viable alternative for temporal video annotation.* The Resource Description Framework [World Wide Web Consortium 2004a], its companion ontology language, RDF Schema [Brickley et al. 2004], and the Jena[1] toolkit was used for the implementation of the OntoLog model. Though primarily

[1] http://jena.sourceforge.net/

aimed at describing web pages and other kinds of documents, RDF's flexibility, expressiveness and simplicity has been shown to be well suited for video metadata as well.

## 9.2 Further work

There are a lot of things in this thesis that could be investigated further. More tests could be run, and more functionality and expressive power could be added to the model and the tools. However, one should be careful not to lose sight of the vision of the OntoLog project: to create a system that, though powerful and flexible, is sufficiently simple and user-friendly for actual users to actually use it.

Possibilities for further work include:

- OWL, Web Ontology Language [World Wide Web Consortium 2004b], is an extension of RDFS, with increased expressiveness and power. OWL Lite is an OWL variant with nice properties regarding reasoning and decidability, and Jena 2 supports this specification. It would be interesting to see how a more powerful ontology language and reasoning system would affect OntoLog.
- One of the most common arguments against systems like OntoLog is that creating accurate semantic annotations manually is too time-consuming. Support for this task, ranging from relatively simple tools such as those described in section 6.3.1, through segmentation algorithms based on visual or aural features of the media, to automatic classification of intervals based on face, voice or language recognition, should be investigated. Automatic classification, such as is used in VoiceGraph [Oard 1997], seems particularly appropriate for the ontology-based stratification of OntoLog.
- Further testing of OntoLog in realistic circumstances, in different domains and for different purposes, should be performed. It would be interesting to investigate how OntoLog could handle
    - very large amounts of data,
    - very big ontologies,
    - very long videos, or
    - videos of very different length,

  and how this would affect the user interfaces and tools.
- Changes and/or additions to the conceptual model could be considered. For instance, *instants* (like intervals without duration) could be added, to accommodate users who need to log instantaneous events such as cuts. This would of course also affect the user interfaces and tools. Supporting different kinds of relationships between intervals and concepts (analogous to the basic and primary contexts of VideoSTAR [Hjelsvold et al. 1995b]), might also be worth looking into.

- More advanced use of ontologies could be imagined, e.g. to interpret natural language queries, or to perform automatic interpretation of Symbols and Indexes. It is imaginable that a query “show me some footage about peace” could be answered by a shot of white doves (regardless of the original intention of the dove shot) with the help of an ontology containing the knowledge that doves are a Symbol of peace.
- Versioning, reuse and long-term development of ontologies should be studied.
- Collaborative annotation could be considered. This includes model support for authorship, trust, rights and uncertainty, as well as tool support for collaboration, import of annotations and author-aware visualisation.

# A Ontologies

*Ontology* is one of the buzzwords in many fields of computer science these days. It is considered a key element of the next generation of computer systems, where intelligent agents roam the hyperspace, gathering and analysing information for their masters; where search engines can actually answer questions, instead of just digging up a set of likely documents that may or may not answer the question; where information systems and databases actually understand the data they manage, and are able to reason about it. However, there is some confusion about what the term "ontology" really means, and how it differs from related terms like taxonomy and thesaurus.

The oldest usage of "ontology" is as an abstract noun denoting a branch of metaphysics or philosophy concerned about the nature and relations of being [Merriam-Webster Incorporated 2004] – what different kinds of things there are in the world and how they are associated. [Merriam-Webster Incorporated 2004] further defines the corresponding concrete noun – an ontology – somewhat tersely as a "particular theory of about the nature of being or the kinds of existents". This is a bit intangible for practical purposes, though.

The most quoted definition of an ontology is given by Gruber [Gruber 1993]: "An ontology is a specification of a conceptualization". This of course begs the question, what exactly is a conceptualization, and how is it specified? Borst [Borst 1997] has elaborated on Gruber's definition by saying that an ontology is a *formal* specification of a *shared* conceptualization, and Studer et al. [Studer et al. 1998] explains this in more detail: A conceptualization is an abstract model of some phenomenon in the world, obtained by identifying the relevant concepts of that phenomenon. This is similar to a conceptual model or database schema, except that an ontology provides "a domain theory, and not the structure of a data container"[1] [Horrocks et al. 2003]. The ontology should be explicit and formal – that is, machine-readable and with explicit definitions of the types of concepts and their use. For it to be of any use, it should be shared – that is, it should capture consensual knowledge, knowledge that is not private to some individual, but accepted by a group.

Though some will argue that an ontology need not be formally specified, this is a necessity for it to be of much use in computer science, since a computer cannot reason efficiently about it otherwise. Hence, ontologies in computer science are often specified in logic-based languages, with clearly defined semantics. That an ontology should represent shared knowledge is primarily convention and common sense – ontologies are typically used for information exchange, to make certain that two or more parties (be they humans or computers) are using a common,

[1] It should be noted, however, that conceptual and logical models in theory also have nothing to do with (physical) data container structure; they are merely logical data organisations.

well-defined vocabulary. However, an ontology may be useful for an isolated individual as well, as a framework for organising and constructing knowledge about a domain.

But what exactly does such a conceptualization look like? And how is it related to the perhaps more common terms "thesaurus" and "taxonomy"? Pidcock [Pidcock 2003] defines these concepts as follows, more or less corresponding with the discussion on semantic depth in [Dörr et al. 2001]:

- A *controlled vocabulary* (or dictionary) is an explicitly enumerated list of terms, controlled by some definition authority. It may be specified formally, but often the semantics are implicitly understood by the vocabulary's users.
- A *taxonomy* is a hierarchy of vocabulary terms. There may be different kinds of parent-child relationships, e.g. whole-part, genus-species, type-instance or class-subclass, but many taxonomies are limited to one kind of relationship.
- A *thesaurus* is a network of vocabulary terms. In addition to hierarchical relations, thesauri also use associative lexical relations like synonym and antonym.
- An ontology is (or may be) all of the above, but typically with the ability to capture more complex relationships between concepts. A *formal ontology* is a controlled vocabulary specified in a formal ontology language that provides a grammar for using vocabulary terms to express something meaningful within a specified domain of interest. The grammar provides formal constraints on how the vocabulary terms can be used together.

The distinctions are blurry; some say that thesauri are extensions of taxonomies, while others considers them more or less equivalent; some consider taxonomies and thesauri special cases of ontologies [Swartout et al. 1996], while others consider such usage of ontology a perversion of the term.

According to [Corcho et al. 2001], an ontology consists of five kinds of things:

- Classes (or concepts) are abstractions of anything about which something is said. They can represent classes of concrete thing, e.g. the class of elephants, but also classes of abstract things like tasks and actions. They are organised in a taxonomy with a subclass relation.
- Instances are specific members of classes. For example, Dumbo the elephant is an instance of the class of elephants.
- Relations represent a type of interactions between entities of the domain. The hierarchical subclass relation of the class taxonomy is an example of a binary relation, as is the "is-a" (or "instance-of") relationship between Dumbo and the elephant class.

- Functions are a special case of relations, where the last element of the relation is unique, given the preceding elements. A mother may have many children, but a child has only one mother (though some definitions of "mother" may relax this constraint); thus, the "has-mother" relation is a function, while "has-child" is not.
- Axioms are statements that can used to define the meaning of ontology components, to put constraints on how terms may be related, to deduce new information from or to verify the consistency of the ontology.

This list is not universally agreed upon, though. Some do not bother to separate functions from other relations, while others introduce *attributes* as meaning properties or relations with values from the primitive type domain (strings, numbers etc.). Some use the word *constraints* instead of axioms, or as a generalisation of it.

In the end, it is difficult to find a common, terse and concrete definition of what an ontology is. But it is possible to establish a few characteristics of ontologies, that hold for all definitions and are sufficient for the treatment of ontologies in this thesis:

- An ontology can define a set of classes of objects, relate the classes in a partial ordering of subclasses, and define individual objects as instances of one or more classes.
- An ontology can define properties for describing elements of the ontology and relating elements to each other with formally specified semantics, and put constraints on how these properties can be used.

# B OntoLog RDF Schema

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:dc="http://purl.org/dc/elements/1.1/">

  <rdf:Description
        rdf:about="http://www.idi.ntnu.no/~heggland/ontolog/ontolog-
        schema#">
    <dc:title>The OntoLog schema</dc:title>
    <dc:description>The schema for Jon Heggland's Ontolog
        system</dc:description>
    <dc:creator>Jon Heggland</dc:creator>
    <dc:date>2003-05-12</dc:date>
  </rdf:Description>

  <!-- Classes -->

  <rdfs:Class rdf:ID="Ontology">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-
        schema#Resource"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="MediaResource">
    <rdfs:label>Media Resource</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-
        schema#Resource"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="Interval">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-
        schema#Resource"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="Project">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-
        schema#Resource"/>
  </rdfs:Class>

  <!-- Properties -->

  <rdf:Property rdf:ID="hasMnemonic">
    <rdfs:comment>Stores the mnemonic character for concept-based
        logging</rdfs:comment>
    <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-
        schema#Literal"/>
  </rdf:Property>

  <rdf:Property rdf:ID="hasInterval">
    <rdfs:label>has interval</rdfs:label>
    <rdfs:domain rdf:resource="#MediaResource"/>
    <rdfs:range rdf:resource="#Interval"/>
    <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/terms/hasPart"/>
  </rdf:Property>

  <rdf:Property rdf:ID="from">
    <rdfs:label>From</rdfs:label>
    <rdfs:domain rdf:resource="#Interval"/>
    <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-
        schema#Literal"/>
  </rdf:Property>

  <rdf:Property rdf:ID="to">
    <rdfs:label>To</rdfs:label>
    <rdfs:domain rdf:resource="#Interval"/>
    <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-
        schema#Literal"/>
  </rdf:Property>
```

```
  <rdf:Property rdf:ID="relatesTo">
    <rdfs:comment>Signifies that an interval relates to a
        concept.</rdfs:comment>
    <rdfs:domain rdf:resource="#Interval"/>
    <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-
        schema#Resource"/>
    <rdfs:subPropertyOf
        rdf:resource="http://purl.org/dc/terms/references"/>
  </rdf:Property>

  <rdf:Property rdf:ID="orderOfChildren">
    <rdfs:comment>Specifies the order of the children of Ontologies and
        Concepts</rdfs:comment>
    <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
        ns#Seq"/>
  </rdf:Property>

  <rdf:Property rdf:ID="usesOntology">
    <rdfs:domain rdf:resource="#Project"/>
    <rdfs:range rdf:resource="#Ontology"/>
    <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/terms/requires"/>
  </rdf:Property>

  <rdf:Property rdf:ID="hasMediaResource">
    <rdfs:domain rdf:resource="#Project"/>
    <rdfs:range rdf:resource="#MediaResource"/>
    <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/terms/hasPart"/>
  </rdf:Property>

  <rdf:Property rdf:ID="projectVersion">
    <rdfs:domain rdf:resource="#Project"/>
    <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-
        schema#Literal"/>
  </rdf:Property>

</rdf:RDF>
```

# C OntoLog ECDL paper

**Heggland, J.**, "OntoLog: Temporal Annotation Using Ad Hoc Ontologies and Application Profiles," in M. Agosti and C. Thanos (Ed.), *Research and Advanced Technology for Digital Libraries (ECDL)*, 2002, p.118–128.

# OntoLog: Temporal Annotation Using Ad Hoc Ontologies and Application Profiles

Jon Heggland

Department of Computer and Information Science, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway
jon.heggland@idi.ntnu.no

**Abstract.** This paper describes OntoLog, a prototype annotation system for temporal media. It is a Java application built to explore the issues and benefits of using ontologies, application profiles and RDF for temporal annotation. It uses an annotation scheme based on hierarchical ontologies, and an RDF-based data model that may be adapted and extended through the use of RDF Schema. Dublin Core is used as a default description scheme. The paper also describes an ontology-based logging interface and annotation visualisation, and a web-based searching and browsing system.

## 1 Introduction

The use of temporal, rich media such as video and audio in research, documentation and education benefits significantly from systems providing temporal annotations. High-level, semantic temporal annotations augment the information in the media, adding comments, explanations, references and links. They also act as indices and tables of content, providing access points and summaries.

Though research in the multimedia database field is focussed primarily on efficient storage and delivery of video and audio data, and less on the description mechanisms needed to handle the enormous amounts of information the data provides, several annotation systems have been proposed and implemented e.g. [1], [2], [3]. Likewise, the digital library community and HCI researchers have produced some fine examples of annotation tools ([4], [5], [6], [7], [8]). However, many such systems are tailored to a specific domain and purpose, which means they excel in their area of expertise, but are cumbersome to adapt for different uses. Others are so generic or minimalist that they may be found lacking in preciseness in complex, specialised domains.

I believe there is merit in exploring the issues of creating a lightweight, extensible and adaptable annotation framework. It should be usable without modification for non-demanding applications, but also permit modification and extension of its data model – by the users, for the users. For interoperability and user-friendliness, it should use widely-accepted technologies and metadata standards to achieve this. Since the creation of semantic annotations is potentially very time-consuming, it should also aim to simplify this process.

In this paper, I describe OntoLog, my attempt at creating such a system. It utilises the Resource Description Framework (RDF, [9]) to provide a basic annotation data model that may be extended using RDF Schemas [10]. It provides a fast and simple

M. Agosti and C. Thanos (Eds.): ECDL 2002, LNCS 2458, pp. 118-128, 2002.

interface for logging video, based on the use of ontologies, which is also RDF-based and extensible.

In the next section, I summarise the most important background technologies OntoLog builds upon. Section 3 describes the OntoLog system – its basic data model, its extensibility and adaptability mechanisms, and its user interface. Section 0 discusses preliminary experiences with the system, and section 5 concludes the paper.

## 2 Background

In this section, I describe the most fundamental issues concerning the annotation of temporal media. This is followed by a brief overview of the metadata standards OntoLog uses, and a short discussion of application profiles.

### 2.1 Annotating Temporal Media

Temporal media are media with a time extent – for instance video, animation, speech and music. Annotations are, according to [11], "notes added by way of comment or explanation". Due to the length of temporal media objects (e.g. motion pictures are typically around two hours), and the fact that they may cover many different topics in this time, annotations need to be temporal as well. They must be connected to specific time intervals in the annotated medium.

There are two main temporal annotation schemes: segmented and stratified. The segmented scheme is the oldest and simplest. The idea is to partition the media object into temporal segments, and describe each segment. A common extension of this scheme is to group related, consecutive segments together, creating a multilevel, hierarchical segmentation. This corresponds to the structure of shots, scenes and sequences in television and film production [12]. A problem with this scheme is that it may be hard to determine the most suitable granularity of the segmentation. If a segment is too large, its description will not be completely valid throughout its whole extent. If the segments are too small, descriptions will need to be repeated across consecutive segments, causing duplication of effort. Another problem is that the concepts described in the media object may partially overlap in various ways, again leading to either partially invalid descriptions, or duplication of effort due to fine granularity. All in all, segmented annotations are better suited to describe the structure of a temporal media object than its semantics.

The stratified approach [3] creates layers of descriptions called strata, where each strata describes the temporal occurrences of some concept like a person, place or topic. The intervals in different strata may overlap, so the description of the media object at any given time can be modelled as the union or projection of the strata present at that time. This is a more flexible scheme, but also more complex to implement and create user interfaces for.

Temporal annotations can be created in several ways. Doing it manually is very time-consuming, so much research has been done on algorithms and techniques for producing them automatically. Systems for segmenting video based on editing points or scene analysis are common (e.g. [13], [14], [3]); likewise, audio may be segmented by silence detection or speaker recognition ([15], [16], [17]). Face and speech recognition has been successfully implemented, as has recognition and interpretation of on-screen text. However, this is still rather low-level information, and is to some

degree dependent on domain information (notice the ubiquity of the well-structured news broadcast domain). The extraction of high-level semantics still requires human intervention, as does the augmenting process of adding comments, explanations and references.

### 2.2 RDF and Dublin Core

The Resource Description Framework (RDF, [9]) is a World Wide Web Consortium recommendation; a domain-neutral standard for machine-readable metadata. Its basic data model consists of three object types: resources, properties and statements. A *resource* is anything addressable by a URI [18]. All things described by RDF are resources. A *property* is an attribute or characteristic used to describe a resource. A specific resource together with a property and the value of that property for that resource is a *statement*. These three parts of a statement are called the *subject*, the *predicate* and the *object*, respectively. The object may be a resource or a literal (a simple string or some other primitive datatype).

The RDF Schema recommendation [10] provides a type system for RDF. Among other things, it specifies a mechanism for defining classes of resources and properties, including subclass relationships, and for creating restrictions on what classes of resources each property may be applied to. The typing system is specified in terms of the basic RDF data model, using resources for concepts such as Class and properties for relationships like subClassOf.

The Dublin Core (DC) Element Set ([19], [20]) is a set of fifteen properties designed to cover the most common needs for describing document-like objects. The standard includes attributes such as title, author and date, and specifies their semantics and how they should be used for maximum interoperability. Though Dublin Core it is not related to RDF, the DC schema can be specified in RDF Schema, and metadata using DC may be encoded using RDF syntax.

### 2.3 Application Profiles

Application profiles as a type of metadata schema was first introduced by Heery and Patel in [21]. The background is that when you design a digital library, database or metadata system, there are many different metadata standards to choose from. Typically, however, none fits your need perfectly – they might be too big, too small, to restrictive or too general. Therefore, usual practise is to adapt the standards – select the most relevant elements and ignore the rest, to impose additional restrictions on cardinality and data types and to combine complementing standards.

Application profiles is the formalisation of this adaptation practise. An application profile is defined as a schema that reuse elements from other schemas without introducing new data elements. They may specify permitted values and schemes, and can refine the standard definitions. An application profile might for instance say that the Dublin Core Identifier property must have an ISBN as its value, or that the Coverage property shall only be used to denote a geographical location. Thus, you end up with a schema designed specifically for the task at hand, while still maintaining a fair degree of interoperability. How to specify, use and disseminate such application profiles is an interesting research topic – see e.g. [22] and [23].

# 3 The OntoLog System

OntoLog is a media annotation tool that uses ontologies or classification schemes to create and access stratified temporal annotations, and provides application profile functionality through integration of different metadata schemas. The main objectives during the development of OntoLog were:

- To explore the issues of enabling flexible, user-defined description schemes and application profiles.
- To simplify production, access and understanding of semantic, temporal annotations by using user-defined ontologies and vocabularies.
- To experiment with a novel visualisation of temporal annotations.

In this section, I first discuss OntoLog's ontology-based annotation scheme, followed by a description of its basic data model. Then, I discuss how OntoLog takes advantage of various metadata standards, before I conclude describing the user interface – the logging interface, the annotation visualisation, and the ontology-enabled, web-based search and browsing system.

## 3.1 Ontology-Based Annotation

Many video indexing systems, e.g. [4], [6], [24], segment the video according to topics, scenes or editing points, and annotate the segments with free-text transcripts, descriptions or keywords. OntoLog uses a different approach. It creates intervals that are unconstrained in that they may overlap freely with each other, like the stratified model presented in section 2.1. Each interval is connected to a concept or term in an ontology, vocabulary or classification scheme. Additionally, the intervals (and the concepts) may be described with properties according to various user-specified description schemas. This approach is arguably more powerful, since it may be used to implement the more restricted schemes – the intervals may be created so as to segment the media object, and annotated using various properties. Even a hierarchical segmentation scheme may be implemented by relating each layer of intervals to a shot, scene or sequence concept, and connecting the intervals between layers using a "part of" property.

Basing the indexing around a structured set of terms has many advantages, as noted by Weinstein [25]. It allows for easy and exact analysis and statistics on the length, occurrences and frequency of each term, within and among media clips. This is useful for domains such as ethnography, anthropology or other application where analysis of behaviour documented on video or audio is common. It also facilitates browsing and searching, since the ontology may be used as an index or catalogue. Compared to using free text descriptions and keywords, it is less prone to uncertainty due to misspellings and use of slightly different words to express the same concept. With support for equivalence relations and "similar term"-relations, ontologies become even more powerful. Additionally, having a standard set of terms to use makes it easier for a group of indexers to produce consistent and interoperable descriptions, and speeds up the annotating process.

OntoLog organises the concepts in hierarchical ontologies, which is also an important point. Using categories for logging video typically produces a lot of categories – the experimental project described in [7] used about 80. A flat list of this

size is quite unwieldy, but arranged in a hierarchy, it is far easier to use. It also allows for easy aggregation of annotations and customisable level of detail during both logging and browsing.

### 3.2 Basic Data Model

Fig. 1 is an UML diagram showing the basic skeleton of OntoLog's data model. The Media Resource element represents the digital media objects, e.g. MPEG files. Each Media Resource contains an unbounded number of Intervals, with start time and end time. By default, no restrictions are put upon the temporal ordering of the intervals, so they may freely overlap.

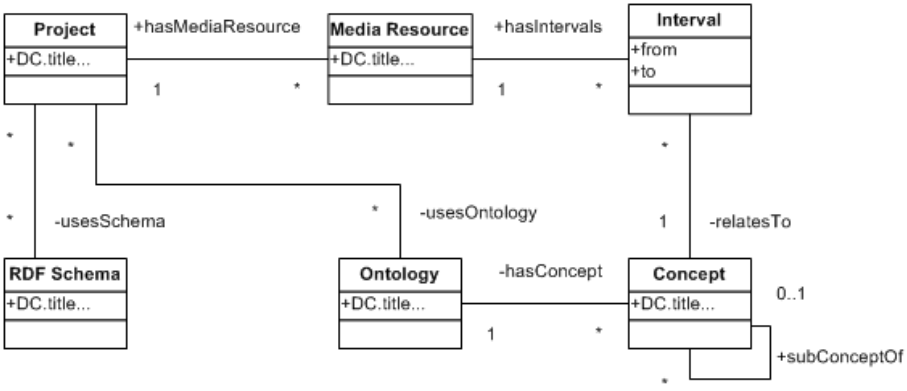


**Fig. 1.** OntoLog's Basic Data Model

There are two principal annotation methods in OntoLog. The main intended annotation mechanism is based on that each of the intervals is connected to a Concept. Concepts may represent terms, topics, persons, places, events – anything that it is desirable to mark the presence of in the media object. Concepts are organised into hierarchical Ontologies, using a relation with subclass or subset semantics. This creates a stratified annotation scheme, augmented by the hierarchical organisation of the strata.

Another complementing annotation mechanism – suggested by the "DC.title…" attributes in the diagram – is that the intervals and concepts (and indeed any data element in OntoLog) may be described with arbitrary properties, selected or defined by the user. The RDF Schema objects determine what properties are available, what classes of resources they may be applied to, and what types of values each property may have.

The Project class has several roles in the data model. It groups a set of media resources, and also manages the set of ontologies used to describe them. Ontologies are shared between projects, to enable reuse, consistency and interoperability. The perhaps most interesting use of the Project class, is its role as application profile. It manages a set of RDF Schemas, which determine how the rest of the data in the project may be described. The project may also adapt the schemas for the purpose at hand, by adding titles and descriptions, and possibly restricting the domains and ranges of the properties. Naturally, RDF Schemas may also be shared between projects.

### 3.3 Application of Metadata Standards

The data model described above is built on the RDF data model, where information is represented as statements consisting of subject, predicate and object. In the diagram, the classes act as subjects and objects, while the associations and attributes are predicates. OntoLog uses the Jena framework [26] to manipulate RDF data, storing the data in a generic relational database. Indeed, all the data in OntoLog, including the schema defining its basic data model, is expressed in RDF. This makes it easy to incorporate other standards and mechanisms that are RDF compatible, and it enables OntoLog to be indexed by RDF search engines and augmented with other RDF tools.

OntoLog is able to import and interpret RDF Schemas. The resources and properties specified in the schemas are integrated into OntoLog's user interface for adding and editing properties, thus creating a simple yet fairly powerful mechanism for extending and adapting OntoLog's capabilities. The needs of different domains and purposes are accommodated through use of different schemas. As OntoLog supports an unlimited number of schemas per project, there is no need for a single schema to encompass all possible metadata requirements.

As a default, OntoLog uses the Dublin Core Element Set 1.1 [20] as its description schema. Dublin Core was chosen because it is a widely accepted standard, and its set of attributes is by design applicable in most domains and suitable for various purposes. As Dublin Core by default puts no constraints on what kinds of resources its elements may be applied to, all kinds of entities in OntoLog – projects, media resources, ontologies, concepts – may be described with DC properties. The DC title property is used (if available) to represent the resources visually in the user interface. For RDF Schema classes and instances of the RDF Property class, the RDF Schema label property (taken from the schemas in which the classes and properties are defined) is used instead.

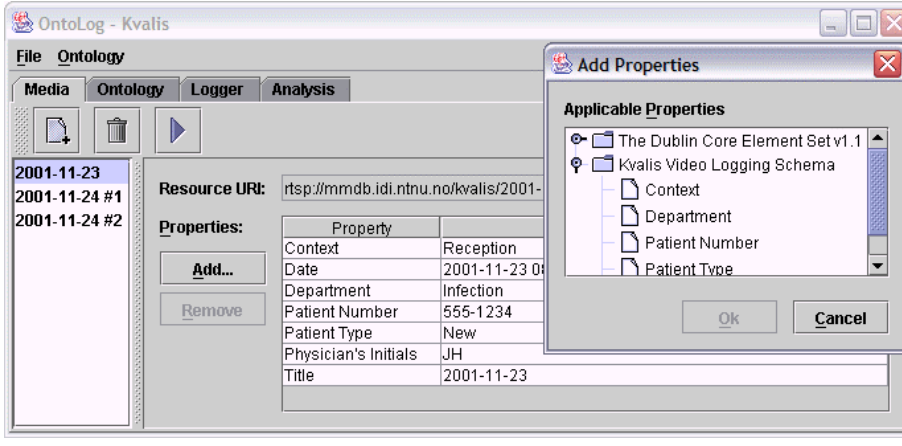


**Fig. 2.** Media Resources with Properties

Fig. 2 shows an example of how a media resource may be annotated with properties drawn from different schemas. This is taken from a project evaluating the use of electronic journals in hospitals through analysing video recordings of medical consultations. The properties of the selected video (titled "2001-11-23") is shown; the

Date and Title properties are from Dublin Core, while the rest of the properties are from a schema designed for the project. The "Add Properties" dialog shows how applicable properties are presented: as a two-level tree structure, the roots being the names of the schema the properties are defined in.

### 3.4 The User Interface

OntoLog's user interface is designed to be simple and fast to use. Due to its use of the RDF data model, describing and managing projects, media resources and ontologies is consistent and straightforward. Importing and using other RDF Schemas is transparent and seamless, since OntoLog already uses RDF Schema for its basic data model and its default description schema, Dublin Core.

**Real-Time Logging Interface.** In logging mode, each concept in the ontologies can be clicked on and off during playback, thus creating intervals linked to the concepts. A keyboard shortcut can be defined for each concept, further simplifying the process. Informal studies of users logging video with OntoLog have shown that with a reasonably small number of relevant concepts, only one or a few passes through the clip is necessary for an adequate set of annotations.

Fig. 3 shows the logger interface logging a video from the video analysis project mentioned in section 3.3. The tree on the left shows the ontology used to annotate the video; the underlined characters in the concept titles indicate the keyboard shortcuts. The timeline display on the right shows the annotations connected to each concept.

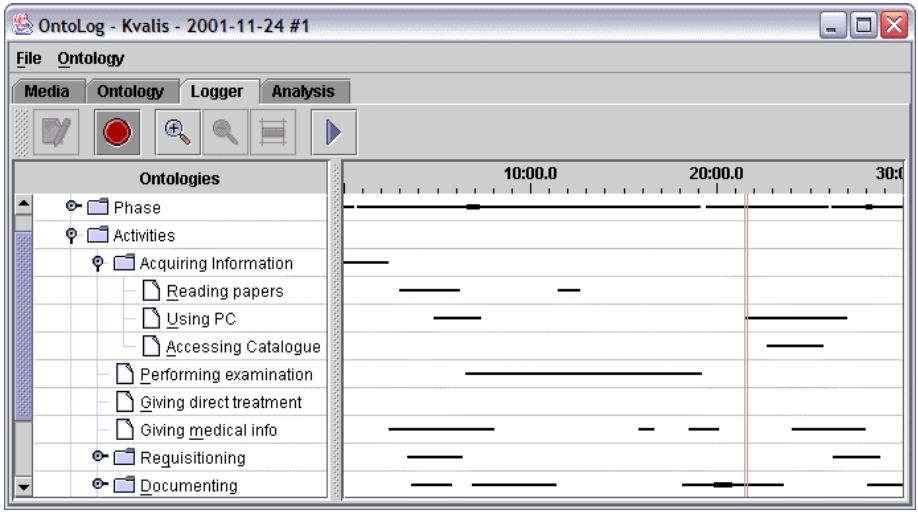


**Fig. 3.** OntoLog's Logger Interface

Once created, the annotations may be edited by direct manipulation. The start and end times of the intervals can be modified by click-and-drag, and intervals can be reclassified to other concepts by dragging them. Properties can be added, edited and removed through a pop-up dialog box.

**Visualisation of Stratified, Hierarchical Annotations.** OntoLog exploits the hierarchical organisation of the ontologies in its visualisation of the annotation intervals. The ontologies are displayed in a collapsible tree structure, while the corresponding annotations are shown as horizontal lines next to each concept in a timeline display. If a concept is collapsed – that is, hiding its descendants – the corresponding set of lines representing annotations is similarly collapsed, displaying the union of all the annotations linked to the concept and its descendants, thickening the line according to how many intervals overlap at any given time. This provides a nice visual summary of the annotations – with most or all of the concepts collapsed, the display shows how thickly the media is annotated, and which concept subtrees are most important. By expanding and collapsing subtrees, users can concentrate on the concepts that are most relevant to the task at hand, and hide non-relevant information.

Fig. 3 illustrates this hierarchical aggregation in some of the collapsed concepts, but in Fig. 4 the visualisation of all the intervals connected to the Activities concept and its subconcepts shows it better. Here, it is plain to see that some activity is "active" throughout the video (as there are no gaps in the line), and that a lot of different activities take place near the start of the video, where the line is very thick.


**Fig. 4.** Collapsed Concept with Lots of Annotations

**Web-Based Searching and Browsing Interface.** As part of a project on streaming digital media, a web-based searching and browsing system called OntoLog Crawler has been developed. It takes advantage of the simplicity and uniformity of the RDF subject-predicate-object data model to provide a simple and consistent browsing interface to the data in OntoLog. It dynamically creates web pages describing RDF resources by listing the RDF statements they are used in, both as subject, predicate and object. Each of the terms in each statement are hyperlinks leading to pages describing them and their statements in the same way. Fig. 5 shows the page describing the MediaResource resource, one of the classes in OntoLog's basic data model (cf. Fig. 1).

OntoLog Crawler also takes advantage of the semantics provided by OntoLog's fundamental RDF Schema. The pages describing media resources and intervals contain a media player configured to play the relevant media clip, and the ontology and concept pages shows the structural organisation of the concepts.

The browsing system requires some familiarity with the RDF data model. The search system does not have this drawback. Given a search term, it searches the properties describing media resources, concepts and intervals, and presents the results as lists of hyperlinks leading to media clips. It can also join the set of clips produced by a search, combining them into a seamless presentation using SMIL [27]. The search system also utilises the semantics of the data model: If a concept is considered a “hit”, the concepts in the subtree below it are also considered “hits”, and all the intervals related to them are included in the search result.

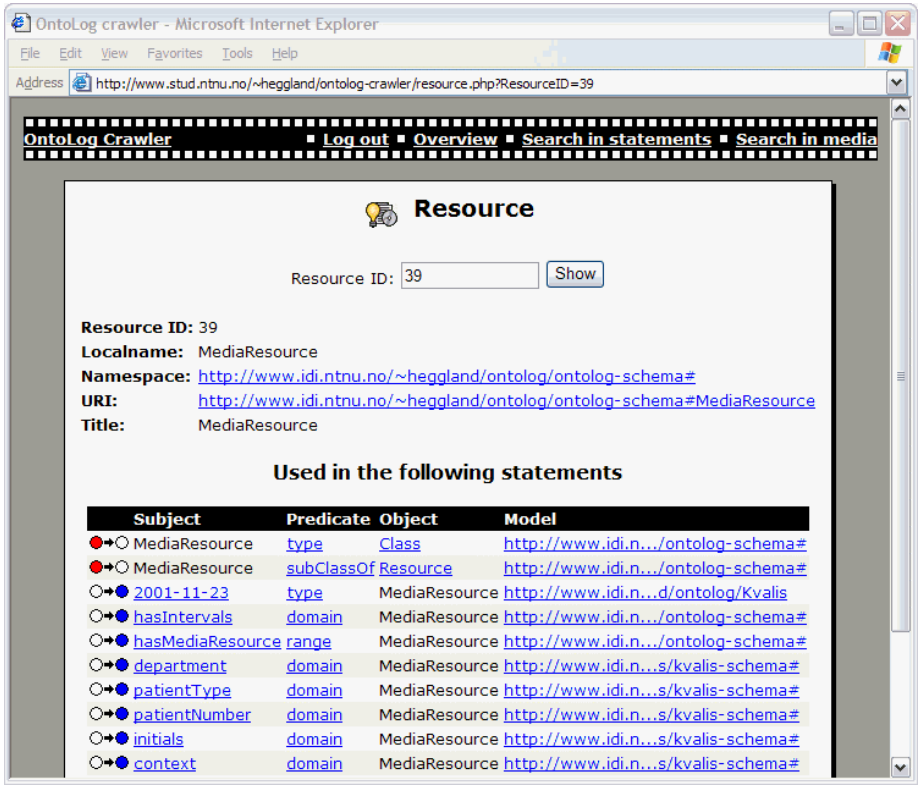


**Fig. 5.** Browsing OntoLog RDF Data

## 4 Experiences

OntoLog has during development been informally evaluated as a possible replacement for Qualitative Media Analyzer (QMA [28]) in a project analysing video recordings of medical consultations. This evaluation has mainly been concerned with the user interface, which was found to be better than QMA's in several ways. Particularly the editing and rearrangement of annotation intervals is easy and intuitive in OntoLog. The logging interface was also considered good, especially the complementary methods of keyboard shortcuts and mouseclicks for turning concepts on and off during logging. On the other hand, the use of Java for media playback makes rapid navigation through video somewhat sluggish.

The extensibility and adaptability functions in OntoLog were evaluated to a lesser degree, due to the small scale of the project. A small RDF Schema for describing the videos was constructed (cf. Fig. 2). The integration of this into OntoLog was very smooth, but it is a drawback that OntoLog as yet does not provide a graphical user interface for creating such schemas – they have to be written by hand in RDF/XML. This is not a problem if standard schemas are used, as they already exist in RDF/XML form, but OntoLog should cater to the needs of projects needing or wanting to create their own schemas without much RDF knowledge as well.

## 5 Conclusion and Further Work

OntoLog is a system for logging and annotating video and audio swiftly and accurately, using user-modifiable combinations of ontologies, RDF schemas and metadata standards. Its purpose is to function as a testbed for exploring the issues concerning the use of application profiles, ontologies and RDF for temporal annotations. In its current incarnation, OntoLog's extensibility and adaptability mechanisms are simple yet powerful, its use of RDF enables it to interoperate with the semantic web, search engines and other RDF-enabled tools with ease, and its user interface is clean, fast and friendly.

However, OntoLog is by no means a finished tool; the version of OntoLog described in this paper is merely a first increment. For instance, its support for RDF Schemas is a bit rudimentary, and its ability to adapt them for a specific application leaves something to be desired. Similarly, the ontology handling can be extended to handle more advanced models, e.g. DAML+OIL [29]. There is also work in progress to apply the SESAM searching approach [30] to OntoLog.

**Acknowledgements.** I would like to thank Hallvard Lærum for ideas and input on the capabilities of OntoLog, particularly the ontology-based logging interface and the visualisation of the intervals, and for evaluating the system. Also, thanks to my advisor Roger Midtstraum and my colleague Jon Olav Hauglid, for feedback on OntoLog and critical reviews of this paper; and to Per Håkon Meland and Jørgen Austvik for their work on OntoLog Crawler. This work is supported by Accenture.

## References

1. Correia, N. and T. Chambel. *Active Video Watching using Annotation*. in *The seventh ACN international conference on Multimedia*. 1999. Orlando, FL USA.
2. Mele, F. and G. Minei, *Digital Video Management for Heterogeneous and Distributed Resources.* IEEE Multimedia, 2001. **8**(3): p. 30-38.
3. Chua, T.-S., L. Chen, and J. Wang, *Stratification Approach to Modeling Video.* Multimedia Tools and Applications, 2002. **16**(1/2): p. 79-97.
4. Weher, K. and A. Poon. *Marquee: A Tool For Real-Time Video Logging*. in *Human factors in computing systems: "celebrating interdependence"*. 1994. Boston, MA USA.
5. Carrer, M., *et al.*, *An Annotation Engine for Supporting Video Database Population*, . 1996, Multimedia Communications Laboratory, Boston University: Boston.
6. Hunter, J. and R. Iannella. *The Application of Metadata Standards to Video Indexing*. in *European Conference on Digital Libraries*. 1998. Crete.
7. Cohen, J., M. Withgott, and P. Piernot. *Logjam: a tangible multi-person interface for video logging*. in *CHI 99 conference on human factors in computing systems*. 1999. Pittsburgh, Pennsylvania: ACM Press.
8. Hunter, J. and D. James. *The Application of an Event-Aware Metadata Model to an Oral History Project*. in *European Conference on Digital Libraries*. 2000. Lisbon, Portugal.
9. Lassila, O. and R.R. Swick, *Resource Description Framework (RDF) Model and Syntax Specification,* http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/. The World Wide Web Consortium 1999.
10. Brickley, D. and R.V. Guha, *Resource Description Framework (RDF) Schema Specification 1.0,* http://www.w3.org/TR/2000/CR-rdf-schema-20000327/. The World Wide Web Consortium 2000.

11. Merriam-Webster, *Merriam-Webster's Online Collegiate Dictionary*, http://www.m-w.com/dictionary.htm. 1998.
12. Monaco, J., *How to Read a Film*. 1981: Oxford University Press.
13. Arman, F., *et al. Content-based Browsing of Video Sequences*. in *ACM Multimedia*. 1994. San Francisco, USA: ACM Press.
14. Foote, J., *et al. An intelligent media browser using automatic multimodal analysis*. in *ACM Multimedia*. 1998. Bristol, UK: ACM Press.
15. Hindus, D., C. Schmandt, and C. Horner, *Capturing, Structuring and Representing Ubiquitous Audio.* ACM Transactions on Information Systems, 1993. **11**(4): p. 376-400.
16. Arons, B., *SpeechSkimmer: A System for Interactively Skimming Recorded Speech.* ACM Transactions on Computer-Human Interaction, 1997. **4**(1): p. 3-38.
17. Whittaker, S., *et al., Jotmail: a voicemail interface that enables you to see what was said*, in *CHI Letters*. 2000. p. 89-96.
18. Berners-Lee, T., *et al., Uniform Resource Identifiers (URI): Generic Syntax*, http://www.ietf.org/rfc/rfc2396.txt. 1998.
19. Weibel, S., *Metadata: The Foundations of Resource Description*, in *D-Lib Magazine*. 1995.
20. DCMI, *Dublin Core Element Set, Version 1.1 - Reference Description*, http://www.dublincore.org/documents/dces/. 1999.
21. Heery, R. and M. Patel, *Application profiles: mixing and matching metadata schemas*, in *Ariadne*. 2000.
22. Baker, T., *et al., What terms does your metadata use? Application profiles as machine-understandable narratives.* Journal of Digital Information, 2001. **2**(2).
23. Hunter, J. and C. Lagoze. *Combining RDF and XML Schemas to Enhance Interoperability Between Metadata Application Profiles*. in *WWW10*. 2001. Hong Kong.
24. Hunter, J. and J. Newmarch. *An Indexing, Browsing, Search and Retrieval System for Audiovisual Libraries*. in *European Conference on Digital Libraries*. 1999. Paris, France.
25. Weinstein, P.C. *Ontology-Based Metadata: Transforming the MARC Legacy*. in *ACM Digital Libraries*. 1998. Pittsburgh, USA.
26. McBride, B., *Jena: Implementing the RDF Model and Syntax Specification*, http://www-uk.hpl.hp.com/people/bwm/papers/20001221-paper/. 2000.
27. Ayars, J., *et al., Synchronized Multimedia Integration Language (SMIL 2.0)*, http://www.w3.org/TR/smil20/. The World Wide Web Consortium 2001.
28. Skou, C.V., *Qualitative Media Analyzer*, http://www.cvs.dk/qma.htm. 2002.
29. Harmelen, F.v., P.F. Patel-Schneider, and I. Horrocks, *Reference Description of the DAML+OIL (March 2001) Ontology Markup Language*, 2001.
30. Hauglid, J.O. and R. Midtstraum. *SESAM - Searching Supported by Analysis of Metadata*. in *ACM Symposium on Applied Computing*. 2002. Madrid, Spain.

# D OntoLog Crawler ACM SIGIR paper

**Austvik, J., Meland, P. H. and Heggland, J.**, "Using Ontologies and Semantic Networks with Temporal Media," Semantic Web Workshop, ACM SIGIR, Toronto, July 28 – August 1 2003.

# Using Ontologies and Semantic Networks with Temporal Media

Per Håkon Meland
Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
per.h.meland@sintef.no

Jørgen Austvik
Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
jorgen.austvik@extend.no

Jon Heggland
Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
jon.heggland@idi.ntnu.no

## ABSTRACT

Use of digital video and audio in computer systems and on the Web is increasing more and more. It is imperative to organise such media with informative metadata, otherwise it quickly becomes an overwhelming task for the users to find the information they are looking for.

Utilising an existing data model based on semantic networks, ontologies and the Resource Description Framework (RDF), we have developed a practical application for searching and navigating in metadata for temporal media. The user interface has two layers of abstraction. This gives the novice user easy access both to the basic metadata and playback functionality for the temporal media. To locate detailed information and relations, the advanced user can perform more in-depth explorations of the semantic network by taking advantage of the lower layer.

## Categories and Subject Descriptors

H.5.0 [**Information Interfaces and Presentation**]: General; H.3.0 [**Information Storage and Retrieval**]: General; E.2.m [**Data Storage Representations**]: Miscellaneous

## General Terms

Management, Experimentation

## Keywords

Ontology-based Information Retrieval, Metadata Representation for Temporal Media, Semantic Networks, Resource Description Framework

## 1. INTRODUCTION

The use of temporal, rich media such as video and audio in computer systems is becoming more and more prevalent. The technologies for producing, processing and storing digital video and audio are maturing and becoming more affordable day by day. The possible uses of digital video are numerous; surveillance, broadcasting, entertainment, healthcare, education, documentation and research, to mention a few. Thus, the amount of digital video available on the web and in other computer systems is rapidly increasing. Describing and organising these huge amounts of data for retrieval is a necessity, and a challenge.

Digital video is represented in a computer system as a time-dependent series of bitmap images, along with a synchronised audio track. It is extremely hard to accurately and swiftly extract useful knowledge from this representation - who and what is shown in the video, what it is about and what is happening - so it is necessary to use more high-level, structured metadata to describe the semantic content of videos. Traditionally, video archive systems have used fixed, proprietary metadata schemas; either designed for a particular purpose and thus unsuited for reuse, or with very general and bland semantics - jacks of all trades, but masters of none.

In the recent years, the related technologies of semistructured data [1], the Semantic Web [32], semantic networks [27] and the Resource Description Framework (RDF) [31] have arisen as an exciting new paradigm for using metadata. RDF provides a metadata framework that is easily extensible and self-describing, yet with well-defined semantics and a very simple basic data model. Using this kind of technology, metadata systems can be tailored to specific domains and purposes, yet still remain interoperable and capable of being accessed by standard tools and search systems.

User-friendly and powerful search interfaces are perhaps the most important part of a metadata system, and it is not obvious how such a system should be designed to take maximum advantage of this way of organising metadata. In this paper, we present one possible solution: The browsing and searching interface OntoLog Crawler, based on the OntoLog video annotation system [12].

In the next section, we briefly discuss the various technologies OntoLog Crawler builds upon. Section 3 focuses on the OntoLog system, the context of OntoLog Crawler. The

Crawler itself, its design and construction, is described in section 4, while section 5 presents our experiences with it. Section 6 discusses further work, and section 7 concludes the paper.

## 2. BACKGROUND

In this section, we briefly present the ideas and technologies that are the foundations of OntoLog Crawler. We describe the issues of annotating temporal media, and how ontologies and semantic networks can be used in this regard. Then we present some of the technology OntoLog Crawler is based on and some related work.

### 2.1 Annotating Temporal Media

When putting a data object in an archive, it is prudent to annotate it with different kinds of descriptions; title, date, authors, topics etc. This makes it easier to perform searches and retrieve the content later. This is especially important for non-textual data objects like images and sound, as their semantics are not readily understandable by a computer. The differences in how for instance sound is experienced by a human being (as music, noise, conversation; different speakers, topics and moods) and a computer (as a time-dependent sequence of numbers representing varying air pressure) create a **semantic gap** that must be bridged by annotations that can be understood by both man and machine.

Temporal media (media with a time dimension, like video and audio) place particular demands on annotation systems. A video segment typically deals with several different topics, places, events or people at different times, so the annotations must be temporal as well - connected to specific intervals in the annotated medium. The alternative - connecting the annotations to the video as a whole - is unsatisfactory for several reasons. As each annotation, e.g. a topic indicator, will not necessarily be valid for the entire video, the user has to manually browse through it to find what he or she is interested in. This is very time-consuming; far more so, in fact, than doing the same in a non-temporal medium, since you have to spend a significant amount of time watching the medium at its predetermined playback speed to grasp its content. It also requires the entire media object to be transferred to the user, instead of just the relevant parts. Since digitization of video and audio create huge amounts of data, this is of course undesirable. Last but not least, temporal annotations are useful for explaining and augmenting the media with comments and explanations. They can be used to create a non-temporal visual index to the medium, providing context information and enabling swift browsing and navigation.

### 2.2 Ontologies and Temporal Annotations

A fairly common scheme for annotating temporal video, is to establish a list of people, places, events, topics or objects that occur in the video, and indicate the temporal intervals in which each is present. This is called stratified annotations [26], as this list of concepts and their intervals form independent strata or layers of meaning. The complete description of the video at any instant is found by taking the cross-section of the strata at that point in time.

This scheme can be improved by organising the concepts and their strata in ontologies, instead of in flat lists. Here, we use "ontology" to mean a formal specification of a set of terms, concepts or classes; their attributes and relations to each other, often in the form of a hierarchy of classes, subclasses and instances (like in object-oriented systems) or wider/narrower-term relationships (like in thesauri). Such ontologies have long been used in artificial intelligence and knowledge systems; they are an emerging technology in many other different fields of computer science, and will play a crucial role in realising the dream of the Semantic Web.

Organising the concepts in a hierarchical ontology has many benefits. Since relationships between the concepts are explicit and specific, inference, analysis and aggregation is straightforward to perform. For instance, the OntoLog application [12] utilises the hierarchical organization to create a visual representation of the annotations where the strata belonging to the subtree of a concept can be aggregated visually, hiding the subtree without losing much information. This makes it easier to handle large sets of concepts, and gives the user control over the level of annotation details. Similarly, OntoLog Crawler takes advantage of the ontology semantics when performing queries, as described in section 4.

Ontologies are very useful for information exchange, so this is one of their primary uses in computer systems. They can be independent of language and application, which makes them ideal for standardising terms and definitions. Predefined ontologies for many different domains can already be found publicly available on the Internet; using them saves time, and increases the possibilities of annotation reuse. An ontology created for a specific domain can therefore be portable between applications in the same domain.

### 2.3 Semantic Networks and RDF

Semantic networks [27] are irregular data structures that express relations between terms. They are directed graphs of nodes connected by arcs. The arcs denote the relations between the nodes, and the nodes are informative objects. This is a very expressive and flexible data structure, but this very flexibility makes them hard to handle by non-experts, and difficult to make user interfaces for.

Using ontologies to structure semantic networks alleviates this problem. An ontology makes for a reasonably stable, intuitive and well-defined tree structure of domain knowledge, to which all sorts of information may be added through the openness of the semantic network model.

The **Resource Description Framework** (RDF) [31] is a standard for machine-readable metadata, built on the semantic network model. Its basic data model consists of the **statement**: a triplet of subject, predicate and object, corresponding to source node, arc, and target node in a semantic network. Figure 1 shows a simple RDF model (adapted from [17]); the web resource **http://www.stud.ntnu.no/~heggland/ontolog-crawler** has two creators, Meland and Austvik. RDF is designed to be domain independent, and its sister standard, RDF Schema [5], defines mechanisms for defining types of nodes and arcs, and putting constraints on how they may be connected. Like ontologies, RDF is playing a big part in realising the Semantic Web.

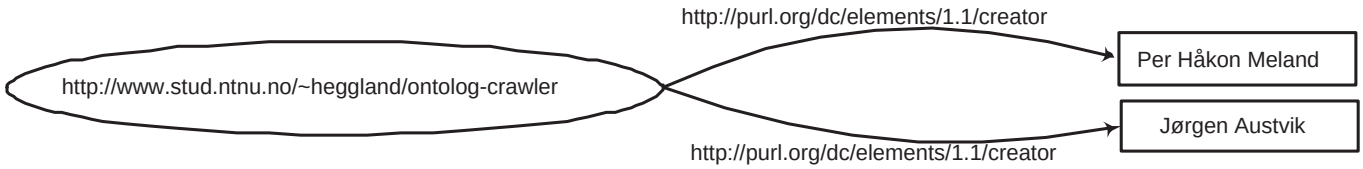


**Figure 1: A simple RDF model.**

## 2.4 Related Work

Semantic annotations for video databases is not a new idea. A number of data models have been proposed, e.g. [20], [2], [13], [30], [6], [15], [7], [9]; some have even been implemented. Tools for creating MPEG-7 annotations are maturing ([21], [28], [14]), though some find MPEG-7 too restrictive and complex ([19], [18]). However, less effort have been put into designing powerful and user-friendly systems and interfaces for accessing this information. The systems mentioned above typically provide a complex query language and/or a list or visualisation of the annotations of a single video, with dubious usability. Information access in databases storing the semantics of multimedia is significantly different from information access in traditional databases, as noted by [18] and [24], and it is this challenge that this work aims to explore.

# 3. THE ONTOLOG SYSTEM

OntoLog Crawler is part of a modular system, where each module has its dedicated tasks. Standardised interfaces between the modules enables a module to be easily replaced, or other modules added as a supplement. Currently, the OntoLog System consists of five main modules, as shown in figure 2: the Annotation Tool, the RDF Storage, the Temporal Media Source, and the Search and Browse Server and its Client. In this section, we will give a brief description of these modules, and section 4 will take a closer look at OntoLog Crawler, our Search and Browse Server.

## 3.1 Annotation Tool

The role of the Annotation Tool is to produce and edit annotations - RDF-based metadata describing the contents of the material stored in the Temporal Media Source. This role is filled by the OntoLog application, which is described in detail in [12].

The annotations are created according to the OntoLog data model framework, shown in figure 3. It consists of **ontologies**, composed of a directed acyclic graph of **RDF classes** and their instances; temporal **intervals**, each belonging to a **media resource** and related to a member of an ontology, thus establishing a stratified annotation scheme; **schemas**, defining the properties and relations used to describe the data; and **projects**, which group a set of ontologies, media resources and schemas for convenience. Dublin Core [8] is used as a default schema for new projects, and other schemas can be created or imported. In addition to these framework classes and relations, other can be defined by the user, according to his or her needs, through the construction of schemas and ontologies. These extensions are seamlessly integrated in the OntoLog system, as everything is defined homogeneously using the RDF model.

The OntoLog application depends on manual annotations, but it can be replaced or supplemented by other automatic or semi-automatic tools that uses the same data model.

## 3.2 RDF Storage

For the RDF Storage module, we are currently using a MySQL relational database. Possible alternatives include other SQL databases such as Oracle, plain text files on a file system, or a database specifically built for XML/RDF. However, MySQL have the advantages that there are RDF programming libraries available for it (OntoLog uses Jena [16]), and it is easy to interface with a web server (as OntoLog Crawler does).

## 3.3 Temporal Media Source

The Temporal Media Source can be realised with numerous sub-modules and with different combinations of these. A video server is the best alternative because it has the ability to stream and initiate playback at any time in the media stream by using the Real Time Streaming Protocol (RTSP) [25] and the Session Description Protocol (SDP) [10]. You can also use regular HTTP-streaming (progressive download) with most video servers. Other alternatives are web servers, a local file system or a video database. Of course, you can combine many such sources, e.g. by using an Oracle 9i video database as data source for a Helix video server.

We have used the Helix Server (formerly known as the RealSystem Server) from RealNetworks, and the Darwin Streaming Server (DSS) version 4 from Apple to stream video and audio. The latest version of DSS supports MPEG-4, which is rapidly becoming the new de facto digital video standard. DSS is OpenSource and runs on most Microsoft Windows platforms and various UNIX clones.

## 3.4 Search and Browse Client

The Search and Browse Client is simply a standard web browser, e.g. Microsoft Internet Explorer, Netscape Navigator or Opera. Most users are already familiar with web browsers, and they are usually preinstalled on most computers. To be able to playback the temporal media, a SMIL-capable multimedia plugin such as RealPlayer or QuickTime player must be installed as well. SMIL (Synchronized Multimedia Integration Language [4]) is an open markup standard defined by the W3C, very similar to HTML. By using SMIL it is simple for the Server to instruct the Client to merge together segmented media clips from various sources. Transitions between the clips are automatically generated on the fly, and a subtitle beneath the video indicates which interval is currently being played.

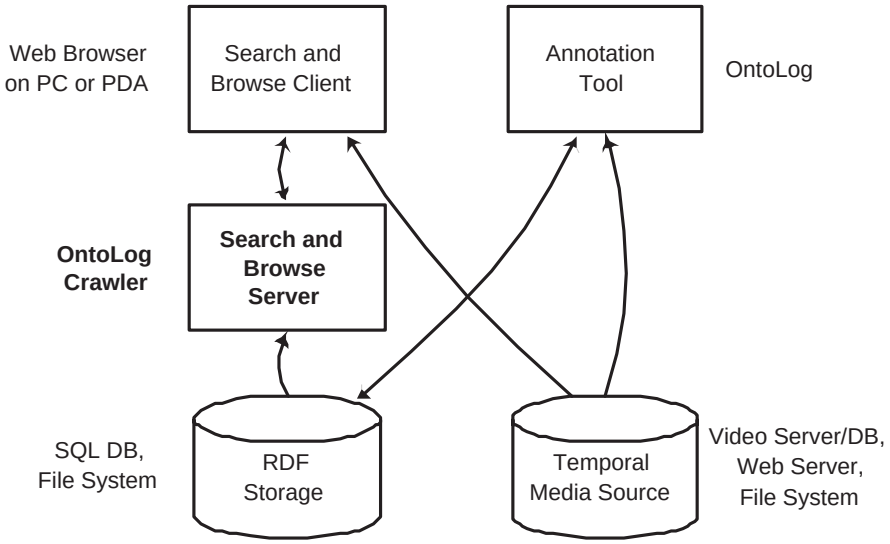


**Figure 2: An overview of the OntoLog system.**

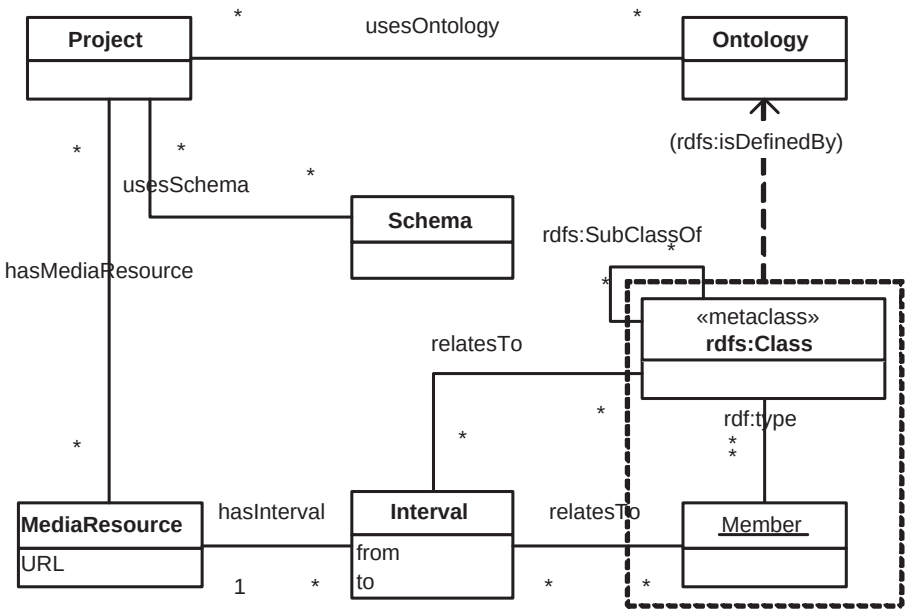


**Figure 3: The OntoLog data model.**

The demands on the client are so simple that it can easily run on a PDA (e.g. an iPAQ) with a wireless network connection. The Client sends queries and navigation requests to the Search and Browse Server, and receives web pages and references to media objects in the Temporal Media Source.

# 4. ONTOLOG CRAWLER

For the Search and Browse Server we have built OntoLog Crawler [3], which is a powerful and user-friendly module for searching and browsing metadata stored in a semantic network. This tool is to be used after the temporal media have been annotated, and makes the stored information available to various types of users. They may be interested in the metadata and the temporal media for research, entertainment or educational purposes, to mention a few. Today, there are not many availible applications that have the ability to access a semantic net for practical usage, and even fewer that that takes advantage of domain knowledge. The next section describes an example scenario that OntoLog Crawler is well suited for, while the subsequent sections describe different aspects of OntoLog Crawler's design, functionality and user interface.

## 4.1 Scenario: Patient Consultations

In a pilot project at St.Olavs Hospital in Trondheim, a video camera records the patient-doctor conversation, the examination, the doctor's computer screen and more. The recordings are then annotated with OntoLog according to a specific ontology, where notes and various types of information can be added to the intervals of the video. This data can be used as a basis for different kinds of research, for example examining the work process during the consultation. Figure 4 shows the playback of sample patient consultation video intervals in OntoLog Crawler.

Related to the introduction of **Electronic Patient Record** (EPR) in hospitals, it can be interesting to study how much time the doctor uses to locate data with the traditional paper based patient record compared to new EPR systems. The example ontology in figure 8 shows some conceps that can be used to annotate videos of patient consultations related to EPR usage. It is based on the ontology used in the project at St. Olavs Hospital, but simplified for this scenario. Concepts such as **Pictures** and **Test results** corresponds to activities related to examining and analysing electronic pictures and test results.

With OntoLog Crawler, you can quickly get an overview of the intervals where either an EPR or a paper based record is being used. Then, you can select one of them and compare which method is most efficient, either by looking through the intervals (as shown in figure 4) or the time lengths of the intervals.

It is recommended to use more general and standardised ontologies if the annotations are going to be used for more than one purpose. These can be supplemented or extended by specially designed ontologies tailored for specific needs.

## 4.2 Abstraction Layers

The OntoLog Crawler uses a layered model to represent the semantic network, offering two different perspectives on the data. The bottom layer consist of a generic and low level representation of the RDF-data. The top layer has knowledge of the OntoLog specific data model, and therefore has knowledge about the multimedia domain. You can go directly from the domain-specific representation of the data into the low-level RDF-data, but not the other way around. Just as one can not construct a sentence from single words without knowing the grammatical constraints of the language, one can not go from the RDF layer to the multimedia layer.

It is possible to search and browse both independent and dependent of the multimedia domain. This gives the novice users easy access to both the basic metadata and playback of the temporal media. Advanced users get the ability to really dig into the semantic network and acquire all the details.

## 4.3 Browsing

The browsing feature in OntoLog Crawler is useful for exploring relatively small datasets, or when you know approximately where to look. However, as the amount of data increases, it is easy to get lost in the semantic network.

Figure 5 shows a low-level RDF-perspective of the **Electronic Patient Record** concept resource, and all the statements where it occurs. The statements consist of subjects, predicates and objects, which in turn are resources of their own (except that objects may be string literals instead). Every part of the statements is a hyperlink that can be followed for more detailed information, making it easy to go to related terms. If you follow the links under **Model**, you will get a list of all the statements in the RDF model.

Figure 6 shows a high-level RDF-perspective of the same concept as in figure 5. Here the properties, related media intervals and the concept's position in the ontology is displayed. If the user follows the **Play merged resultset** link, a virtual media document is created by merging all the intervals (concatenating them while keeping in mind that they may overlap). Figure 4 shows an example of such a document.

## 4.4 Searching

As for browsing, the search facility in OntoLog Crawler can be used from two perspectives. The low-level search returns the statements that matches the query, which is useful when looking for specific resources. The high-level search utilises the semantics of the OntoLog data model, and presents the results in a more user-friendly way. Instead of statements, it returns a list of intervals, mediaresources and concepts that matches the query.

The searching and browsing features compliment each other in an advantageous way: First you can search to locate some interesting data, and then you can browse through the related information. For example, if you search for **Electronic Patient Record**, you get all the intervals where the doctor uses EPR. You may then browse the related metadata and find information such as the date/time of the consultation.

In OntoLog Crawler, rollup and drilldown in the search result is realised with a combobox that show all the generalizations (super concepts) and all first-level specializations


**Figure 4: Embedded playback of merged video intervals.**

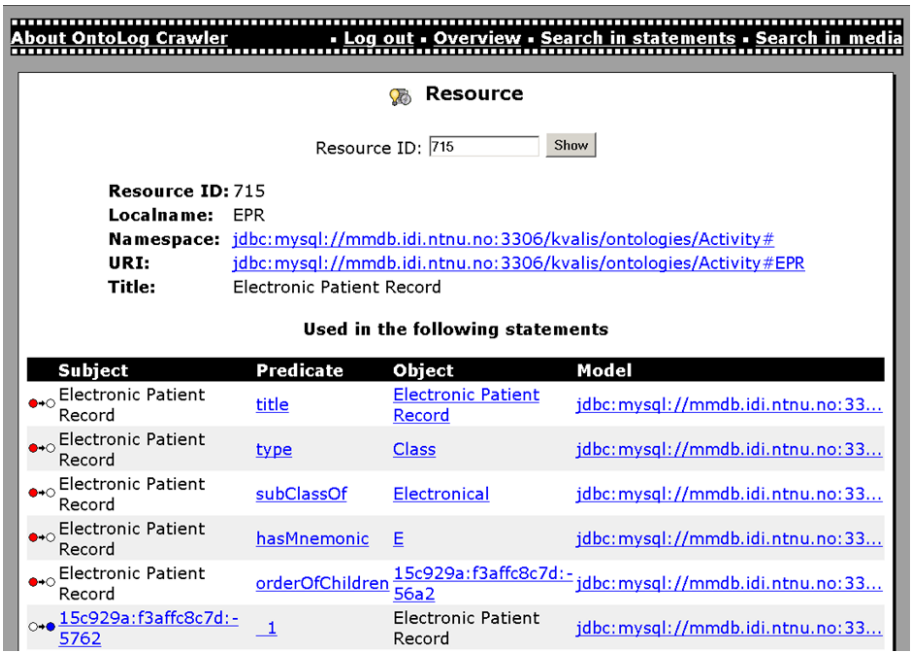


**Figure 5: Low abstraction layer view of a concept.**

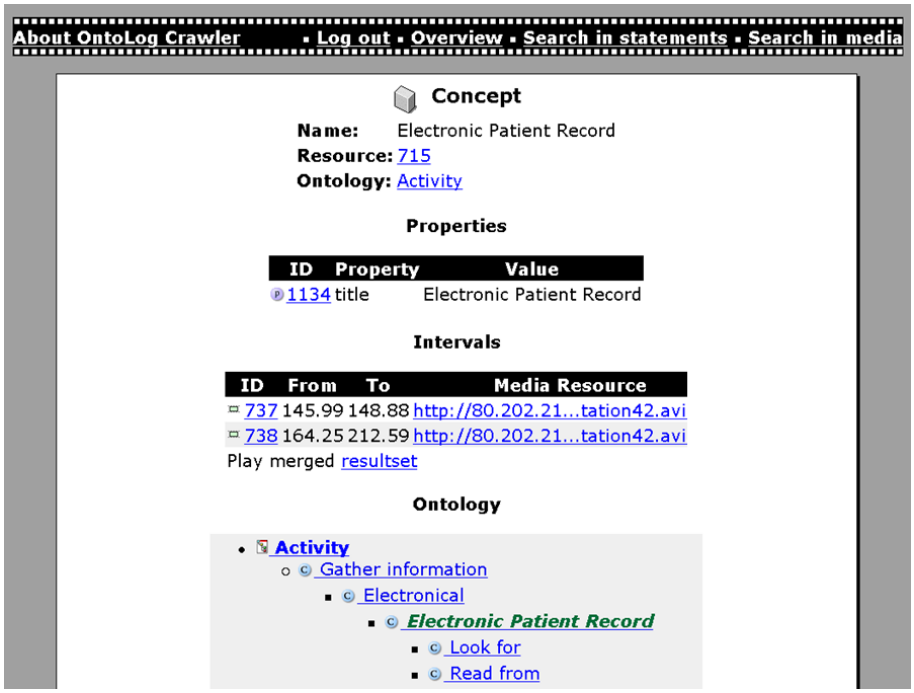


**Figure 6: High abstraction layer view of a concept.**

(sub concepts) of the concept that matches a search query. Figure 7 shows the result of a search for **patient record**. By selecting another concept in the combobox, a page with information about that concept and all related media intervals is displayed.

Figure 8 shows another result from a search for **patient record**. Here the whole ontology is displayed, and both the concepts and sub-concepts that match the search query are highlighted. This gives the user a better and more visual overview over the result and the related concepts. The user can click on the concept to get more information and display its intervals. If the ontologies are large, this view may occupy a great deal of screen space.

## 4.5 Visualisation

OntoLog Crawler generates SMIL code that the Client uses as a playback-control with the Temporal Media Source. The use of SMIL makes OntoLog Crawler independent of the Temporal Media Source, and enables a lot of possibilities for the presentation.

OntoLog Crawler has the ability to export the RDF-data into the **Visualization of Compiler Graph** (VCG) language [23]. VCG was constructed to detect errors in compilers, but works nicely for visualising semantic networks. By using viewer software such as **aiSee**, you can zoom, pan and get a good overview of the nodes and the relations between them.

# 5. EXPERIENCES

In this section we will share some of the experiences from our work. This includes general remarks about OntoLog Crawler and the other modules, and also findings from some

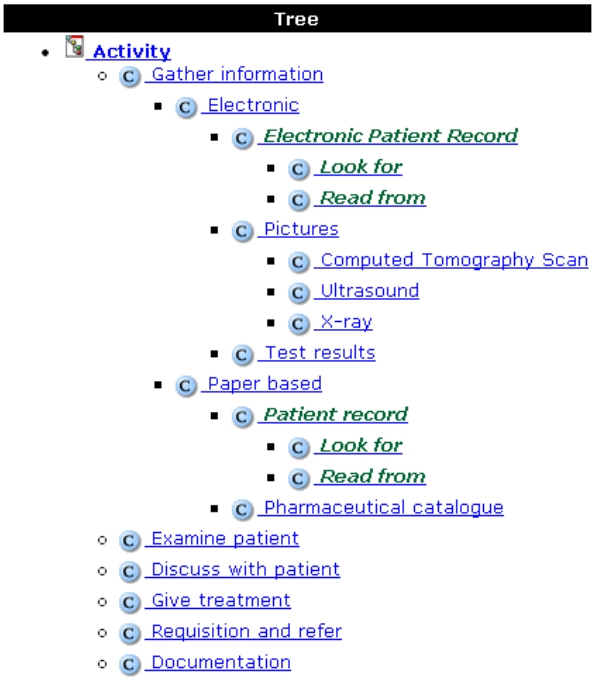


**Figure 8: Sample ontology for annotating medical consultations.**


**Figure 7: With a combobox it is easy to perform rollup and drilldown in the ontology.**

experiments and a user evaluation.

## 5.1 The User Interface and Abstraction Layers

The structure of a semantic network can be very difficult to understand for inexperienced users. On the other hand, it contains a lot of useful information which experienced users are able to dig into. By making the user interface layered, with two abstraction layers of the semantic data, both user groups may utilise the information stored in the web. We should mention that the bottom layer is well suited for debugging other applications that use Jena for RDF storage.

A usability test of OntoLog Crawler showed that even though experienced computer users understood the principles of semantic networks well enough, it was a bit difficult to perform practical tasks. For example, some users got lost in the semantic network when trying to find detailed metadata about intervals. This shows that it is important to build a user interface that hides the underlying data structure for the average user. Another conclusion from the user test was that inaccurate or incorrect annotations of metadata may mislead or confuse the user to a great extent. In practice, the data may easily be lost if the metadata is wrong - just like a book that is placed on the wrong shelf in a library.

One can add an additional abstraction layer on top of the multimedia layer to create a specially suited user interface for a particular domain. This would result in a more user-oriented interface, such as a video archive for patient consultations integrated with the EPR-system.

Weinstein [29] points out numerous advantages when using ontologies with predefined concepts instead of keywords or text. It is possible to perform analysis and inference on the data, as in **Searching Supported by Analysis of Metadata** (SESAM) [11], where the application uses statistics computed on the search result to produce filters to narrow down the results (reduce the noise in the result set). It is also possible to generalise and specialise the queries, and to integrate different ontologies by creating mappings between them.

## 5.2 SMIL Usability

The various media players/plugins support the core SMIL elements very well, and this shows that this type of technology is mature enough to be used in various web applications. The text-based and open nature of SMIL makes it ideal for server-side scripting where the content is dynamic.

## 5.3 Practical Experiments

The OntoLog system is module-based, so to achieve higher performance for each module (and thus the whole system), it was advantageous to deploy the modules over more than one physical server. This way we had a dedicated video server, database server and web server. The network connection between these were 100 Mbps Ethernet, and represented no noticeable delay. The system is able to serve a great number of clients, spread all over the world.

Searching in a semantic network can be very time consuming. In our solution, the data structure itself is stored in an ordinary SQL-database with tables generated by Jena. These tables are generic for RDF-data, so it is almost impossible to look directly into the tables and find useful information. For example, when searching for metadata concepts described in the OntoLog data model, you have to create a SQL query consisting of fourteen table joins. Our studies have shown that the search time more than doubles if the number of statements in the database doubles. This non-linear behaviour is caused by the increased number of connections in the semantic network.

Search performance can be optimised by constructing domain specific indices over the RDF-structure, but this demands knowledge of both the domain and the application. This is of course not consistent with having a general storage structure.

# 6. FURTHER WORK

Even though OntoLog Crawler represents a firm basis for applications like multimedia servers, it should be customised for specific domains. The user interface can be enhanced for the domain, and domain-specific functionality can be added. Examples of this are stored queries and integration with other systems.

The Ana application [22] uses the OntoLog data model for advanced data analysis, using temporal and logical operators on the media. This makes it possible to search for intervals related to **Norway** and **Curling**, which are followed by **Olympic Gold**. This functionality should be included in the search functionality of OntoLog Crawler.

OntoLog annotates along the time dimension, and does not accept spatial data. If this is incorporated, OntoLog Crawler may also be expanded to take spatial queries. This will make the tool much more useful for services that require object descriptions, for example video surveillance and analysis.

## 7. CONCLUSIONS

We have developed a practical application for searching and browsing in metadata for temporal media. This has enabled us to test out some relevant techniques and technologies. Just like RDF has made exchange and interoperability with the metadata possible, SMIL has bridged the gap between different multimedia servers and players.

Structuring the metadata with domain-specific ontologies reduces the complexity of the semantic network, which ameliorates the performance issues typical for irregular data structures. This is a flexible and applicable data model, where both man and machine are able to interpret and extract useful information. A semantic network may not give much meaning without a suitable ontology.

In OntoLog Crawler the user interface has a layered information representation. This makes browsing in metadata easy for novices and powerful for experienced users. To find relevant information, you can first do a rough search, and then browse your way into the details.

## 8. ACKNOWLEDGMENTS

We would like to thank Rune Rystad for the cooperation on the research related to and development of OntoLog Crawler, Hallvard Lærum for the patient consultations scenario, and Hilde Susanne Hansen and Bjørnar Solevåg at St.Olavs Hospital for their assistance.

## 9. ADDITIONAL AUTHORS

Roger Midtstraum, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway (`roger.midtstraum@idi.ntnu.no`).

## 10. REFERENCES

[1] S. Abiteboul, P. Buneman, and D. Suciu. *From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, San Francicso, California, 2000. 1-55860-622-X.

[2] S. Adali, K. S. Candan, S.-S. Chen, K. Erol, and V. S. Subrahmanian. The Advanced Video Information System: Data Structures and Query Processing. *Multimedia Systems*, 4(4):172–186, 1996.

[3] J. Austvik and P. H. Meland. Lagring, søk og levering av temporale data. Master's thesis, NTNU, June 2002.

[4] J. Ayars, D. Bulterman, A. Cohen, K. Day, E. Hodge, P. Hoschka, E. Hyche, M. Jourdan, M. Kim, K. Kubota, R. Lanphier, N. Layada, T. Michel, D. Newman, J. van Ossenbruggen, L. Rutledge, B. Saccocio, P. Schmitz, and W. ten Kate. Synchronized Multimedia Integration Language (SMIL 2.0), August 2001. http://www.w3.org/TR/smil20/.

[5] D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0: RDF Schema, January 2003. http://www.w3.org/TR/rdf-schema/.

[6] M. Carrer, L. Ligresti, G. Ahanger, and T. D. C. Little. An Annotation Engine for Supporting Video Database Population. *Multimedia Tools and Applications*, 5(3):233–258, 1997.

[7] M. E. Dönderler, E. Saykol, Özgür Ulusoy, and U. Güdükbay. BilVideo: A Video Database Management System. *IEEE MultiMedia*, 10(1):66–70, 2003.

[8] Dublin Core Metadata Initative. Dublin Core Metadata Initative, January 2003. http://dublincore.org/.

[9] N. Fatemi and P. Mulhem. A Conceptual Graph Approach for Video Data Representation and Retrieval. In D. J. Hand, J. N. Kok, and M. R. Berthold, editors, *Advances in Intelligent Data Analysis, Third International Symposium, IDA-99, Amsterdam, The Netherlands, August 1999, Proceedings*, pages 525–. Springer, 1999.

[10] M. Handley and V. Jacobson. SDP: Session Description Protocol (RFC 2327). Technical report, IETF, April 1998.

[11] J. O. Hauglid and R. Midtstraum. SESAM - Searching Support by Analysis of Metadata. In *ACM Symposium on Applied Computing*, March 2002.

[12] J. Heggland. OntoLog: Temporal Annotation Using Ad Hoc Ontologies and Application profiles. In *European Conference on Digital Libraries*, pages 118–128. Springer, September 2002.

[13] R. Hjelsvold and R. Midtstraum. Modelling and querying video data. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 686–694. Morgan Kaufmann, 1994.

[14] IBM. IBM MPEG-7 Annotation Tool. http://alphaworks.ibm.com /tech/videoannex.

[15] F. A. Kokkoras, H. Jiang, I. P. Vlahavas, A. K. Elmagarmid, E. N. Houstis, and W. G. Aref. Smart VideoText: a video data model based on conceptual graphs. *Multimedia Systems*, 8(4):328–338, 2002.

[16] H. Labs Semantic Web activity. The jena semantic web toolkit, January 2003. http://www.hpl.hp.com/semweb/jena-top.html.

[17] F. Manola and E. Miller. RDF Primer, January 2003. http://www.w3.org/TR/rdf-primer/.

[18] F. Nack and L. Hardman. Towards a syntax for multimedia semantics. Technical report, CWI, 2002. http://www.cwi.nl/ftp/CWIreports/INS/INS-R0204.pdf.

[19] F. Nack and W. Putz. Designing annotation before it's needed. In *ACM Multimedia*, pages 251–260, 2001.

[20] E. Oomoto and K. Tanaka. Ovid: Design and implementation of a video-object database system. *TKDE*, 5(4):629–643, 1993.

[21] Ricoh. Ricoh MovieTool. http://www.ricoh.co.jp/src/multimedia/MovieTool/.

[22] R. Rystad. Analyse av Temporale Aspekter i Multimediadatabaser. Master's thesis, NTNU, June 2002.

[23] G. Sander. VCG – Visualization of Compiler Graphs. Technical report, Universitt des Saarlandes, 1995.

[24] S. Santini and R. Jain. Interfaces for Emergent Semantics in Multimedia Databases. In *SPIE Storage and Retrieval for Image and Video Databases VII*, pages 167–175, 1999.

[25] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP) (RFC 2326). Technical report, IETF, April 1998.

[26] T. Smith and N. Pincever. Parsing Movies in Context. In *The 1991 Summer USENIX Conference*, Nashville, USA, 1991.

[27] J. Sowa. *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. Morgan Kaufmann Publishers, January 1991.

[28] Videoto. VIDETO - Video Description Tool. http://www.rostock.zgdv.de /ZGDV/Abteilungen/zr2 /Produkte/videto/index_html_en.

[29] P. C. Weinstein. Ontology-Based Metadata: Transforming the MARC Legacy, June 1998.

[30] R. Weiss, A. Duda, and D. K. Gifford. Composition and Search with a Video Algebra. *IEEE MultiMedia*, 2(1):12–25, 1995.

[31] World Wide Web Consortium. Resource Description Framework (RDF). http://www.w3.org/RDF/.

[32] World Wide Web Consortium. Semantic Web. http://www.w3.org/2001/sw/, May 2001.

# E About Savanta

The research behind Savanta – its conception, design, implementation and evaluation – is a joint effort between Jon Olav Hauglid and Jon Heggland. The purpose was to bring together our two different but related fields of research – Heggland's modelling and visualisation of temporal metadata and Hauglid's iterative, analysis-supported database search – and see what synergy effects could be achieved in the interface between them.

The entire process leading up to the results presented in sections 6.7–6.9 and 7.2 was highly iterative. While we assigned one to be overall responsible for each of the identified parts of the project, the other provided frequent feedback. Thus, by the nature of our close cooperation, it is impossible to completely separate one's contribution from the other's. However, the table below indicates roughly who was the primary contributor to the design (D) and implementation (I) of the various ideas, tools and research described in the Savanta sections:

| | Hauglid | Heggland |
|---|---|---|
| Stored metadata / conceptual model | | DI |
| Derived metadata / analysis | DI | |
| Visualisation | | DI |
| Filtering | DI | |
| Savantoogle | | DI |
| Forms | I | D |
| Interaction model / iterative search refinement | DI | |
| Evaluation | D | I |
| Navigation / web metaphor | DI | DI |

The writing was also done jointly; hence, a substantial part of the material on Savanta is identical in our two theses. The table above is fairly indicative on who did the writing of the various sections as well.

We both agree that we were equal partners and contributors in all major phases of the Savanta project.

Trondheim, June 7. 2004

Jon Olav Hauglid

Jon Heggland

# F Savanta evaluation

This appendix presents details from the Savanta evaluation: the evaluation tasks, the handout given to the test subjects, and the questionnaire results in tabular form.

## Evaluation tasks

Three sets of evaluation tasks were made. Each test subject performed one set of tasks using one interface and all test subjects performed the three sets in the same order. The only think that varied was the order of the interfaces. This was determined randomly.

Each set of evaluation tasks was divided into three parts. The parts were: (A) simple retrieval, (B) complex retrieval and (C) exploration. The tasks were given in Norwegian; an English translation is provided here in parenthesis.

### Set 1:

A1: Omtales HTML i uke 41?
(Is HTML mentioned in week 41?)

A2: Hva sier foreleser om egenrelasjoner?
(What does the lecturer say about recursive relationships?)

B1: Finn eksempler som handler om løkker.
(Find examples about loops.)

B2: Hvor stor andel av forelesningene brukes på praktiske opplysninger?
(How big a part of the lectures is spent on practical information?)

B3: Hva skjer i det første kvarteret av dobbelttime 1 i uke 36?
(What happens in the first quarter of double lecture 1 in week 36?)

C1: Hva er det viktigste hovedtemaet i første tredjedel av semesteret?
(What is the most important main topic in the first third of the semester?)

C2: Hvilket databasetema brukes det mest tid på?
(Which database topic is spent most time on?)

### Set 2:

A1: Omtales JSP i uke 43?
(Is JSP mentioned in week 41?)

A2: Hva sier foreleser om lokale variabler?
(What does the lecturer say about local variables?)

B1: Finn eksempler som handler om objekter.
(Find examples about objects.)

B2: Hvor stor andel av forelesningene brukes på databaser?
(How big a part of the lectures is spent on databases?)

B3: Hva skjer i det første kvarteret av dobbelttime 1 i uke 40?
(What happens in the first quarter of double lecture 1 in week 40?)

C1: Hva er det viktigste hovedtemaet i midterste tredjedel av semesteret?
(What is the most important main topic in the 2nd third of the semester?)

C2: Hvilket HTML-tema brukes det mest tid på?
(Which HTML topic is spent most time on?)

### Set 3:

A1: Omtales databaser i uke 40?
(Is databases mentioned in week 41?)

A2: Hva sier foreleser om evige løkker?
(What does the lecturer say about endless loops?)

B1: Finn eksempler som handler om merkelapper.
(Find examples about tags.)

B2: Hvor stor andel av forelesningene brukes på HTML?
(How big a part of the lectures is spent on HTML?)

B3: Hva skjer i det første kvarteret av dobbelttime 1 i uke 43?
(What happens in the first quarter of double lecture 1 in week 43?)

C1: Hva er det viktigste hovedtemaet i siste tredjedel av semesteret?
(What is the most important main topic in the last third of the semester?)

C2: Hvilket JSP-tema brukes det mest tid på?
(Which JSP topic is spent most time on?)

# Handout

The following Norwegian text was handed out to the test subjects at the beginning of the evaluation. An English translation is given afterwards.

## Evaluering av verktøy for informasjonsgjenfinning

### Introduksjon

Formålet med denne brukerevalueringen er å sammenligne brukbarheten til tre forskjellige verktøy for informasjonsgjenfinning i videodatabaser. Som testdeltaker skal du utføre en del oppgaver i hvert verktøy slik at du kan danne deg en oppfatning av hva du liker og ikke liker. Etterpå får du en del evalueringsspørsmål om hvert av verktøyene. Totalt vil dette ta ca. 1 time. Alle testverktøyene lar deg søke i en videodatabase som inneholder forelesningene til Steinar Line fra faget Informasjonsteknologi grunnkurs høsten 2003.

### Om oppgavene

Noen ganger vet man nøyaktig hva man er ute etter, mens andre ganger er letingen mer tilfeldig. I tillegg kan målet med letingen endre seg underveis. For å se hvordan testverktøyene egner seg til forskjellige måter å søke på, er oppgavene delt opp i tre:

- Enkel informasjonsgjenfinning
- Kompleks informasjonsgjenfinning
- Utforsking

Tre oppgavetyper og tre testverktøy gir ni muligheter. For hver av disse får du 2-3 oppgaver. Rekkefølgen verktøyene evalueres i er tilfeldig, mens rekkefølgen på oppgavetypene alltid er a), b), c).

### Om evalueringen

Evalueringen består av to deler:

*Utføring av oppgaver*

Alle oppgaver vil bli oppgitt på et eget ark. Husk at det er verktøyene som skal evalueres – ikke du. Denne delen tar ca. 45 minutter.

*Evalueringsspørsmål*

Spørsmålene er avkrysningsoppgaver der man gir karakterer fra 1 til 9. Alle spørsmålene er gjengitt under (det vil bli utdelt avkrysningsskjema – dette er bare til orientering):

| | | | | |
|---|---|---|---|---|
| 1 | Bedømmelse – Enkel informasjonsgjenfinning | forferdelig<br>1 2 3 4 5 6 7 8 9 | fantastisk | NA |
| 2 | Bedømmelse – Kompleks informasjonsgjenfinning | forferdelig<br>1 2 3 4 5 6 7 8 9 | fantastisk | NA |
| 3 | Bedømmelse – Utforskning | forferdelig<br>1 2 3 4 5 6 7 8 9 | fantastisk | NA |

| | | | | |
|---|---|---|---|---|
| 4 | Totalbedømmelse av Programmet | forferdelig<br>1 2 3 4 5 6 7 8 9 | fantastisk | NA |
| 5 | Programmet var ? å bruke | vanskelig<br>1 2 3 4 5 6 7 8 9 | Enkelt | NA |
| 6 | Uttrykkskraften var | utilstrekkelig<br>1 2 3 4 5 6 7 8 9 | Akkurat passe | NA |
| 7 | Mengden informasjon presentert på skjermen | for lite eller for mye<br>1 2 3 4 5 6 7 8 9 | Akkurat passe | NA |
| 8 | Teksten som vises på Skjermen | forvirrende<br>1 2 3 4 5 6 7 8 9 | begripelig | NA |
| 9 | Oppgaver kan løses på en rettfram mate | aldri<br>1 2 3 4 5 6 7 8 9 | Alltid | NA |
| 10 | Tilgjengelig hjelp | utilstrekkelig<br>1 2 3 4 5 6 7 8 9 | Akkurat passe | NA |

## Evaluation of tools for information retrieval

### Introduction

The purpose of this evaluation is to compare the usability of three different tools for information retrieval in video databases. As test subject, you are to perform a few tasks in each tool in order to form an opinion of what you like and dislike. Afterwards, you will be given some evaluation questions about each tool. Together, this will take about an hour. All the tools let you search in a video

database containing the lectures of Steinar Line from the course "Information technology, introduction" from the autumn of 2003.

**About the tasks**

Sometimes you know exactly what you're looking for, while sometimes the search is more random. Additionally, the goal of the search may change during the process. To see how the tools are suited to different ways of searching, the tasks are divided into three groups:

- Simple retrieval
- Complex retrieval
- Exploration

Three task groups and three tools results in nine combinations. For each of these, you will perform 2–3 tasks. The order in which the tools are to be evaluated is random, but the order of task groups is always a), b), c).

**About the evaluation**

The evaluation consists of two parts:

*Task performance*

All tasks will be given on a separate sheet of paper. Remember, it is the tools that are to be evaluated – not you. This part will take about 45 minutes.

*Evaluation questions*

The questions are multiple choice; each asking for a grade from 1 to 9. The questions are given below (you will receive a questionnaire sheet – this is just for reference):

[The English versions of the questions (taken from QUIS [Chin et al. 1988]) are given on page 174.]

## Questionnaire results

There were nine test subjects; all randomly chosen computer science students with several years of experience using computers. However, none had any experience with video databases. Each subject was asked ten questions about each tested interface. They could answer using a nine-point scale (1-9). Results are shown in the tables below. Empty cells indicate that the user did not answer the question.

**Savantoogle**

| | User | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Average |
| Simple | 8 | 7 | 5 | 9 | 4 | 9 | 7 | 7 | 5 | 6,8 |
| Complex | 6 | 4 | 2 | 1 | 5 | 3 | 4 | 4 | 4 | 3,7 |
| Exploration | 5 | 2 | 2 | 1 | 3 | 3 | 3 | 2 | 3 | 2,7 |
| Overall | 7 | 6 | 4 | 3 | 4 | 3 | 5 | 3 | 5 | 4,4 |
| Learn to operate | 8 | 8 | 9 | 7 | 5 | 4 | 7 | 8 | 9 | 7,2 |
| Power | 6 | 6 | 2 | 4 | 5 | 4 | 8 | 3 | 3 | 4,6 |
| Amount of info | 6 | 8 | 7 | 4 | 4 | 4 | 6 | 2 | 3 | 4,9 |
| Messages on screen | 8 | 7 | 9 | 9 | 7 | 5 | 6 | 8 | 8 | 7,4 |
| Straight-forward | 5 | 5 | 3 | 3 | 4 | 3 | 5 | 3 | 3 | 3,8 |
| Amount of help | 4 | 5 | 9 | 7 | | 3 | 9 | | | 6,2 |

**Forms**

| | User | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Average |
| Simple | 6 | 7 | 7 | 7 | 4 | 6 | 4 | 5 | 3 | 5,4 |
| Complex | 5 | 6 | 6 | 3 | 5 | 4 | 2 | 5 | 3 | 4,3 |
| Exploration | 4 | 3 | 5 | 4 | 4 | 5 | 1 | 5 | 2 | 3,7 |
| Overall | 6 | 6 | 6 | 5 | 4 | 6 | 4 | 5 | 3 | 5,0 |
| Learn to operate | 9 | 7 | 7 | 9 | 6 | 8 | 5 | 8 | 8 | 7,4 |
| Power | 5 | 5 | 5 | 2 | 4 | 6 | 2 | 6 | 2 | 4,1 |
| Amount of info | 3 | 7 | 7 | 4 | 3 | 6 | 3 | 7 | 2 | 4,7 |
| Messages on screen | 8 | 8 | 8 | 9 | 7 | 7 | 9 | 8 | 7 | 7,9 |
| Straight-forward | 5 | 5 | 4 | 6 | 4 | 5 | 3 | 5 | 3 | 4,4 |
| Amount of help | 3 | 5 | 9 | 8 | | 5 | 4 | | | 5,7 |

**Savanta**

| | User | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Average |
| Simple | 9 | 7 | 8 | 8 | 7 | 8 | 6 | 8 | 8 | 7,7 |
| Complex | 8 | 7 | 7 | 8 | 7 | 7 | 7 | 8 | 8 | 7,4 |
| Exploration | 8 | 5 | 7 | 7 | 7 | 6 | 6 | 7 | 5 | 6,4 |
| Overall | 8 | 7 | 8 | 7 | 7 | 8 | 8 | 7 | 7 | 7,4 |
| Learn to operate | 7 | 7 | 4 | 5 | 6 | 7 | 8 | 6 | 7 | 6,3 |
| Power | 8 | 7 | 8 | 7 | 7 | 8 | 6 | 8 | 7 | 7,3 |
| Amount of info | 8 | 6 | 3 | 4 | 8 | 7 | 7 | 4 | 6 | 5,9 |
| Messages on screen | 8 | 6 | 5 | 7 | 7 | 6 | 7 | 7 | 5 | 6,4 |
| Straight-forward | 8 | 6 | 6 | 5 | 7 | 8 | 6 | 8 | 7 | 6,8 |
| Amount of help | 8 | 7 | 9 | 8 | | 8 | 6 | | | 7,7 |

# References


[Adali et al. 1996] **Adali, S., Candan, K. S., Chen, S., Erol, K. and Subrahmanian, V. S.**, "The Advanced Video Information System: Data Structures and Query Processing," *Multimedia Systems*, **4** (4) August 1996, p.172-186.

[Ahlberg et al. 1992] **Ahlberg, C., Williamson, C. and Shneiderman, B.**, "Dynamic Queries for Information Exploration: An Implementation and Evaluation," *Proceedings of ACM CHI'92 Conference on Human Factors in Computer Systems*, 1992, p.619-626.

[Allen 1983] **Allen, J. F.**, "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, **26** (11) 1983, p.832-843.

[Arman et al. 1994a] **Arman, F., Depommier, R., Hsu, A. and Chiu, M.**, "Content-Based Browsing of Video Sequences," *ACM Multimedia*, 1994, p.97-103. (a)

[Arman et al. 1994b] **Arman, F., Hsu, A. and Chiu, M.**, "Image Processing on Encoded Video Sequences," *Multimedia Systems*, **1** (5) 1994, p.211-219. (b)

[Arons 1997] **Arons, B.**, "SpeechSkimmer: A System for Interactively Skimming Recorded Speech," *ACM Transactions on Computer-Human Interactions*, **4** (1) 1997, p.3-38.

[Arslan et al. 2002] **Arslan, U., Dönderler, M. E., Saykol, E., Ulusoy, Ö. and Güdükbay, U.**, "A Semi-Automatic Semantic Annotation Tool for Video Databases," *SOFSEM Workshop on Multimedia Semantics*, 2002.

[Austvik and Meland 2002] **Austvik, J. and Meland, P. H.**, "Lagring, søk og levering av temporale data", Department of Computer and Information Science, Norwegian University of Science and Technology, June 2002.

[Austvik et al. 2003] **Austvik, J., Meland, P. H. and Heggland, J.**, "Using Ontologies and Semantic Networks with Temporal Media," Semantic Web Workshop, ACM SIGIR, Toronto, July 28 - August 1 2003.

[Baeza-Yates and Ribeiro-Neto 1999] **Baeza-Yates, R. A. and Ribeiro-Neto, B. A.**, *Modern Information Retrieval*, ACM Press / Addison-Wesley, 1999.

[Bechhofer et al. 2000] **Bechhofer, S., Broekstra, J., Decker, S., Erdmann, M., Fensel, D., Goble, C., van Harmelen, F., Horrocks, I., Klein, M., McGuinness, D., Motta, E., Patel-Schneider, P., Staab, S. and Studer, R.**, *An informal description of Standard OIL and Instance OIL*, 2000.

[Bloom 1953] **Bloom, B. S.**, "The Thought Process of Students in Discussion," in S. J. French (Ed.), *Accent on Teaching: experiments in general education*, 1953.

[Boreczky and Rowe 1996] **Boreczky, J. S. and Rowe, L. A.**, "Comparison of Video Shot Boundary Detection Techniques," *Storage and Retrieval for Image and Video Databases (SPIE)*, 1996, p.170-179.

[Borst 1997] **Borst, W. N.**, "Construction of Engineering Ontologies for Knowledge Sharing and Reuse", Centre for Telematics and Information Technology, University of Twente, 1997.

[Brickley et al. 2004] **Brickley, D., Guha, R. V. and McBride, B.**, "RDF Vocabulary Description Language 1.0: RDF Schema", http://www.w3.org/TR/rdf-schema/, February 10 2004.

[Budd 1998] **Budd, T.**, *Understanding Object-Oriented Programming with Java*, Addison-Wesley, 1998.

[Bukauskas 2003] **Bukauskas, L.**, "The Tiger Temporal Database System", http://www.cs.auc.dk/tiger/, 2003.

[Carlis and Maguire 2001] **Carlis, J. and Maguire, J.**, *Mastering Data Modeling: A User-Driven Approach*, Addison-Wesley, 2001.

[Carrer et al. 1997] **Carrer, M., Ligresti, L., Ahanger, G. and Little, T. D. C.**, "An Annotation Engine for Supporting Video Database Population," *Multimedia Tools and Applications*, **5** (3) 1997, p.233-258.

[Chambers 2002] **Chambers, M. L.**, *CD and DVD Recording for Dummies*, 2002.

[Chang et al. 1997] **Chang, S., Chen, W., Meng, H. J., Sundaram, H. and Zhong, D.**, "VideoQ: An Automated Content Based Video Search System Using Visual Cues," *ACM Multimedia*, 1997, p.313-324.

[Chaudri et al. 1998] **Chaudri, V. K., Farquhar, A., Fikes, R., Karp, P. D. and Rice, J. P.**, *Open Knowledge Base Connectivity 2.0.3*, 1998.

[Chin et al. 1988] **Chin, J. P., Diehl, V. A. and Norman, K. L.**, "Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface," *Proceedings of ACM CHI'88 Conference on Human Factors in Computing Systems*, 1988, p.213-218.

[Christel and Martin 1998] **Christel, M. G. and Martin, D.**, "Information Visualization Within a Digital Video Library," *Journal of Intelligent Information Systems*, **11** (3) 1998, p.235-257.

[Christel et al. 1995] **Christel, M. G., Kanade, T., Mauldin, M., Reddy, R., Sirbu, M., Stevens, S. M. and Wactlar, H. D.**, "Informedia Digital Video Library," *Communications of the ACM*, **38** (4) 1995, p.57-58.

[Chua et al. 2002] **Chua, T., Chen, L. and Wang, J.**, "Stratification Approach to Modeling Video," *Multimedia Tools and Applications*, **16** (1) 2002, p.79-97.

[Cohen et al. 1999] **Cohen, J., Withgott, M. and Piernot, P.**, "Logjam: A Tangible Multi-Person Interface for Video Logging," *CHI*, 1999, p.128-135.

[Corcho et al. 2001] **Corcho, O., Fernández-López, M. and Pérez, A. G.**, *Technical Roadmap v1.0*, OntoWeb, 2001.

[DCMI Usage Board 2003] **DCMI Usage Board**, "Overview of Documentation for DCMI Metadata Terms", http://www.dublincore.org/documents/dcmes-qualifiers/, March 4 2003.

[Date 1996] **Date, C. J.**, "Why "The Object Model" Is Not a Data Model," *InfoDB*, **10** (4) 1996.

[Date 2004] **Date, C. J.**, "Temporal Databases," *An Introduction to Database Systems*, 2004, p.727-773.

[Davis 1993] **Davis, M.**, "Media Streams: An Iconic Visual Language for Video Annotation," *Proceedings of the 1993 IEEE Workshop on Visual Languages, August 24-27, 1993, Bergen, Norway*, 1993, p.196-202.

[Denning et al. 1989] **Denning, P. J., Comer, D., Gries, D., Mulder, M. C., Tucker, A. B., Turner, A. J. and Young, P. R.**, "Computing as a Discipline," *Communications of the ACM*, **32** (1) 1989, p.9-23.

[Deregowski 1972] **Deregowski, J. B.**, "Pictorial Perception and Culture," *Scientific American*, **227** (5) 1972, p.82-88.

[Dublin Core Metadata Initiative 2003] **Dublin Core Metadata Initiative**, "Dublin Core Metadata Element Set, Version 1.1: Reference Description", http://www.dublincore.org/documents/dces/, June 2 2003.

[Dublin Core Metadata Initiative 2004] **Dublin Core Metadata Initiative**, "Dublin Core Metadata Initiative Overview", http://www.dublincore.org/about/, 2004.

[Dönderler 2002] **Dönderler, M. E.**, "Data Modeling and Querying for Video Databases", Department of Computer Engineering, Bilkent University, July 2002.

[Dönderler et al. 2003] **Dönderler, M. E., Saykol, E., Ulusoy, Ö. and Güdükbay, U.**, "BilVideo: A Video Database Management System," *IEEE MultiMedia*, **10** (1) 2003, p.66-70.

[Dörr et al. 2001] **Dörr, M., Guarino, N., Fernández-López, M., Schulten, E., Stefanova, M. and Tate, A.**, *State of the Art in Content Standards*, OntoWeb, 2001.

[Engum 2003] **Engum, E. A.**, "Dataverktøy for filmanalyse", Department of Computer and Information Science, Norwegian University of Science and Technology, 2003.

[Findlay and Gilchrist 2003] **Findlay, J. M. and Gilchrist, I. D.**, *Active Vision: The Psychology of Looking and Seeing*, Oxford University Press, 2003.

[Foley et al. 1990] **Foley, J. D., van Dam, A., Feiner, S. K. and Hughes, J. F.**, *Computer Graphics*, Addison-Wesley, 1990.

[Foote et al. 1998] **Foote, J., Boreczky, J. S., Girgensohn, A. and Wilcox, L.**, "An Intelligent Media Browser Using Automatic Multimodal Analysis," *ACM Multimedia*, 1998, p.375-380.

[Fowler 2003] **Fowler, M.**, *UML Distilled*, Addison-Wesley, 2003.

[Galtung 2002] **Galtung, J.**, *Dok1: Galtungs reise*, Norwegian Broadcasting Corporation (NRK), 2002.

[Genesereth and Fikes 1992] **Genesereth, M. R. and Fikes, R. E.**, *Knowledge Interchange Format, Version 3.0 Reference Manual*, 1992.

[Gershon et al. 1998] **Gershon, N., Eick, S. G. and Card, S.**, "Design: Information Visualization," *Interactions*, **5** (2) 1998, p.9-15.

[Gordon 2000] **Gordon, A. S.**, "Using annotated video as an information retrieval interface," *Proceedings of the 2000 international conference on Intelligent user interfaces, January 9 - 12, New Orleans, LA USA*, 2000, p.133-140.

[Gould and Lewis 1985] **Gould, J. D. and Lewis, C.**, "Design for Usability: Key Principles and What Designers Think," *Communications of the ACM*, **28** (3) 1985, p.300-311.

[Gruber 1993] **Gruber, T. R.**, "A Translation Approach to Portable Ontologies," *Knowledge Aquisition*, **5** (2) 1993, p.199-200.

[Hauglid and Midtstraum 2002] **Hauglid, J. O. and Midtstraum, R.**, "SESAM: searching supported by analysis of metadata," *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC)*, 2002, p.418-425.

[Heggland 2002] **Heggland, J.**, "OntoLog: Temporal Annotation Using Ad Hoc Ontologies and Application Profiles," in M. Agosti and C. Thanos (Ed.), *Research and Advanced Technology for Digital Libraries (ECDL)*, 2002, p.118-128.

[Hibino and Rundensteiner 1995] **Hibino, S. and Rundensteiner, E. A.**, "A Visual Query Language for Identifying Temporal Trends in Video Data," *International Workshop on Multimedia Database Management Systems, Blue Mountain Lake, New York*, 1995, p.74-81.

[Hibino and Rundensteiner 1996] **Hibino, S. and Rundensteiner, E. A.**, "MMVIS: Design and Implementation of a Multimedia Visual Information Seeking Environment," *ACM Multimedia*, 1996, p.75-86.

[Hibino and Rundensteiner 1997] **Hibino, S. and Rundensteiner, E. A.**, "User Interface Evaluation of a Direct Manipulation Temporal Visual Query Language," *ACM Multimedia*, 1997, p.99-107.

[Hibino and Rundensteiner 1998] **Hibino, S. and Rundensteiner, E. A.**, "Comparing MMVIS to a Timeline for Temporal Trend Analysis of Video Data," *Proceedings of Advanced Visual Interfaces (AVI '98)*, 1998.

[Hjelsvold 1995] **Hjelsvold, R.**, "VideoSTAR - A Database for Video Information Sharing", Department of Computer Science and Telematics, Norwegian Institute of Technology, 1995.

[Hjelsvold and Midtstraum 1994] **Hjelsvold, R. and Midtstraum, R.**, "Modelling and Querying Video Data," in J. B. Bocca, M. Jarke and C. Zaniolo (Ed.), *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases*, 1994, p.686-694.

[Hjelsvold and Midtstraum 1995] **Hjelsvold, R. and Midtstraum, R.**, "Databases for Video Information Sharing," *Storage and Retrieval for Image and Video Databases (SPIE)*, 1995, p.268-279.

[Hjelsvold et al. 1995a] **Hjelsvold, R., Langørgen, S., Midtstraum, R. and Sandstå, O.**, "Integrated Video Archive Tools," *ACM Multimedia*, 1995, p.283-293. (a)

[Hjelsvold et al. 1995b] **Hjelsvold, R., Midtstraum, R. and Sandstå, O.**, "A Temporal Foundation of Video Databases," in J. Clifford and A. Tuzhilin (Ed.), *Recent Advances in Temporal Databases: Proceedings of the International Workshop on Temporal Databases*, 1995, p.295-314. (b)

[Hjelsvold et al. 1999] **Hjelsvold, R., Liou, S. and Depommier, R.**, "Multimedia Archiving, Logging and Retrieval," in B. Fuhrt (Ed.), *Handbook of Multimedia Computing*, 1999, p.379-402.

[Horrocks et al. 2003] **Horrocks, I., Fensel, D., Broekstra, J., Decker, S., Erdmann, M., Goble, C., van Harmelen, F., Klein, M. S. S., Studer, R. and Motta, E.**, "The Ontology Inference Layer OIL", http://www.ontoknowledge.org/oil/TR/oil.long.html, 2003.

[Hunter 1998] **Hunter, J.**, "The Application of Metadata Standards to Video Indexing," in C. Nikolaou and C. Stephanidis (Ed.), *Research and Advanced Technology for Digital Libraries (ECDL)*, 1998, p.135-156.

[Hunter and Newmarch 1999] **Hunter, J. and Newmarch, J.**, "An Indexing, Browsing, Search and Retrieval System for Audiovisual Libraries," in S. Abiteboul and A. Vercoustre (Ed.), *Research and Advanced Technology for Digital Libraries (ECDL)*, 1999, p.76-91.

[IBM Research 2002] **IBM Research**, "VideoAnnEx Annotation Tool", http://www.research.ibm.com/VideoAnnEx/, 2002.

[IFLA 1998] **IFLA**, "Functional Requirements for Bibliographic Records", http://www.ifla.org/VII/s13/frbr/frbr.htm, 1998.

[International Organization for Standardization 2001] **International Organization for Standardization**, *ISO 8879:1986 Standard Generalized Markup Language (SGML)*, 2001.

[International Organization for Standardization 2002] **International Organization for Standardization**, *ISO/IEC 15938 Information Technology -- Multimedia Content Description Interface*, 2002.

[Jiang et al. 1997] **Jiang, H., Montesi, D. and Elmagarmid, A. K.**, "VideoText Database Systems," *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, 1997, p.344-351.

[Kazman et al. 1996] **Kazman, R., Al-Halimi, R., Hunt, W. and Mantei, M.**, "Four Paradigms for Indexing Video Conferences," *IEEE MultiMedia*, **3** (1) Spiring 1996, p.63-73.

[Kimber et al. 1995] **Kimber, D., Wilcox, L., Chen, F. and Moran, T. P.**, "Speaker segmentation for browsing recorded audio," *CHI 95 Conference Companion*, 1995, p.212-213.

[Kokkoras et al. 2002] **Kokkoras, F. A., Jiang, H., Vlahavas, I. P., Elmagarmid, A. K., Houstis, E. N. and Aref, W. G.**, "Smart VideoText: a video data model based on conceptual graphs," *Multimedia Systems*, **8** (4) 2002, p.328-338.

[Kominek and Kazman 1997] **Kominek, J. and Kazman, R.**, "Accessing Multimedia through Concept Clustering," *CHI*, 1997, p.19-26.

[Lagoze and Hunter 2001] **Lagoze, C. and Hunter, J.**, "The ABC Ontology and Model," in K. Oyama and H. Gotoda (Ed.), *Proceedings of the International Conference on Dublin Core and Metadata Applications*, 2001, p.160-176.

[Lagoze et al. 2000] **Lagoze, C., Hunter, J. and Brickley, D.**, "An Event-Aware Model for Metadata Interoperability," in J. L. Borbinha and T. Baker (Ed.), *Research and Advanced Technology for Digital Libraries (ECDL)*, 2000, p.103-116.

[Le Saux et al. 2003] **Le Saux, B., Grira, N. and Boujemaa, N.**, "Adaptive Robust Clustering with Proximity-Based Merging for Video-Summary," IEEE International Conference on Fuzzy Systems, Saint-Louis, May 2003.

[Li et al. 2000] **Li, F. C., Gupta, A., Sanocki, E., He, L. and Rui, Y.**, "Browsing digital video," *CHI*, 2000, p.169-176.

[Liou et al. 1999a] **Liou, S., Hjelsvold, R., Depommier, R. and Hsu, A.**, "Efficient and Reliable Digital Media Archive for Content-based Retrieval," *Multimedia Systems*, **7** (4) 1999, p.256-268. (a)

[Liou et al. 1999b] **Liou, S., Toklu, C. and Heckrodt, K.**, "VideoTalk: A Collaborative Environment for Video Content Discussion," *Proceedings of the IEEE Internation Conference on Multimedia Computing and Systens*, 1999, p.454-459. (b)

[Litsheim 2003] **Litsheim, S.**, "Romlige beskrivelser i video", Department of Computer and Information Science, Norwegian University of Science and Technology, July 2003.

[Lu 2001] **Lu, G.**, "Indexing and Retrieval of Audio: A Survey," *Multimedia Tools and Applications*, **15** (3) 2001, p.269-290.

[Lærum 2004] **Lærum, H.**, "Evaluation of electronic medical record: A clinical task perspective", Faculty of Medicine, Norwegian University of Science and Technology, March 2004.

[Mackay and Beaudouin-Lafon 1998] **Mackay, W. E. and Beaudouin-Lafon, M.**, "DIVA: Exploratory Data Analysis with Multimedia Streams," *CHI*, 1998, p.416-423.

[Mayo 1933] **Mayo, E.**, *The Human Problems of an Industrial Civilization*, Macmillan, 1933.

[Merriam-Webster Incorporated 2004] **Merriam-Webster Incorporated**, "Merriam-Webster Online Dictionary", http://www.m-w.com/, 2004.

[Metz 1974] **Metz, C.**, *Film Language: A Semiotics of the Cinema*, New York: Oxford University Press, 1974.

[Mills et al. 1992] **Mills, M., Cohen, J. and Wong, Y. Y.**, "A Magnifier Tool for Video Data," *CHI*, 1992, p.93-98.

[Mitchell and Jolley 2001] **Mitchell, M. and Jolley, J.**, *Research Design Explained, Fourth Edition*, Thomson Learning, 2001.

[Monaco 1981] **Monaco, J.**, *How to Read a Film*, New York: Oxford University Press, 1981.

[National Research Council 1994] **National Research Council**, *Academic Careers for Experimental Computer Scientists and Engineers*, Washington, D.C.: National Academic Press, 1994.

[Nielsen 1993] **Nielsen, J.**, *Usability Engineering*, Academic Press, 1993.

[Noldus Information Technology 2003] **Noldus Information Technology**, "The Observer 5.0", http://www.noldus.com/products/observer/index.html, November 2003.

[Oard 1997] **Oard, D. W.**, "Speech-based Information Retrieval for Digital Libraries," *Notes from AAAI Spring Symposium on Cross-Language Text and Speech Retrieval, Stanford University, California*, 1997.

[Oomoto and Tanaka 1993] **Oomoto, E. and Tanaka, K.**, "OVID: Design and Implementation of a Video-Object Database System," *IEEE Transactions on Knowledge and Data Engineering*, **5** (4) 1993, p.629-643.

[Pidcock 2003] **Pidcock, W.**, "What are the differences between a vocabulary, a taxonomy, a thesaurus, an ontology, and a meta-model?", http://www.metamodel.com/article.php?story=20030115211223271, January 15 2003.

[Plaisant et al. 1996] **Plaisant, C., Milash, B., Rose, A., Widoff, S. and Shneiderman, B.**, "LifeLines: Visualizing Personal Histories," *CHI*, 1996, p.221-227.

[Plaisant et al. 1998] **Plaisant, C., Muslin, R., Snyder, A., Li, J., Heller, D. and Shneiderman, B.**, "LifeLines: Using Visualization to Enhance Navigation and Analysis of Patient Records," *Proceedings of the 1998 American Medical Informatic Association Annual Fall Symposium*, 1998, p.76-80.

[Preece et al. 1994] **Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. and Carey, T.**, *Human-Computer Interaction*, Addison-Wesley Publishing, 1994.

[Russel and Norvig 1995] **Russel, S. and Norvig, P.**, *Artificial Intelligence - A Modern Approach*, Prentice Hall, 1995.

[Rust and Bide 2000] **Rust, G. and Bide, M.**, *The <indecs> metadata framework: Principles, model and data dictionary*, 2000.

[Rutledge 2001] **Rutledge, L.**, "SMIL 2.0: XML for Web Multimedia," *IEEE Internet Computing*, **5** (5) September-October 2001, p.78-84.

[Rystad 2002] **Rystad, R.**, "Analyse av kategoriserte metadata i temporale medier", Department of Computer and Information Science, Norwegian University of Science and Tehcnology, June 2002.

[Samet 1990] **Samet, H.**, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, 1990.

[Schoepflin et al. 2001] **Schoepflin, T., Lau, C., Garg, R., Kin, D. and Kim, Y.**, "A Research Environment for Developing and Testing Object Tracking Algorithms," *Proceedings of SPIE Electronic Imaging*, 2001, p.667-675.

[Shneiderman 1996] **Shneiderman, B.**, "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations," *Proceedings of the IEEE Symposium on Visual Languages*, 1996, p.336-343.

[Shneiderman 1997] **Shneiderman, B.**, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Addison-Wesley Publishing, 1997.

[Skou 2003] **Skou, C. V.**, "Qualitative Media Analyzer", http://www.cvs.dk/qma.htm, August 2003.

[Slaughter et al. 1998] **Slaughter, L. A., Oard, D. W., Warnick, V. L., Harding, J. L. and Wilkerson, G. J.**, "A Graphical Interface for Speech-Based Retrieval," *Proceedings of the 3rd ACM International Conference on Digital Libraries*, 1998, p.305-306.

[Snodgrass 1995] **Snodgrass, R. T.**, *The TSQL2 Temporal Query Language*, Kluwer Academic Publishers, 1995.

[Sowa 1984a] **Sowa, J. F.**, "Conceptual graphs for a database interface," *IBM Journal of Research and Development*, **20** (4) 1984, p.336-357. (a)

[Sowa 1984b] **Sowa, J. F.**, *Conceptual Structures: Information Processing in Mind and Machine*, Reading, MA: Addison-Wesley, 1984. (b)

[Staff and Jødahl 2001] **Staff, E. and Jødahl, P.**, *MPEG-7 - bits about the bits*, 2001.

[Stengel 2003] **Stengel, C.**, "Analysis and Representation of Multimedia Data for Issues of Annotation of Temporal Media", Department of Computer and Information Science, Norwegian University of Science and Technology, December 2003.

[Stern 1995] **Stern, P. N.**, "Grounded Theory Methodology: Its Uses and Processes," in B. G. Glaser (Ed.), *Grounded Theory 1984-1994*, 1995, p.29-39.

[Stifelman et al. 2001] **Stifelman, L., Arons, B. and Schmandt, C.**, "The audio notebook: paper and pen interaction with structured speech," *CHI*, 2001, p.182-189.

[Studer et al. 1998] **Studer, R., Benjamins, V. R. and Fensel, D.**, "Knowledge Engineering: Principles and Methods," *Data & Knowledge Engineering*, **25** (1-2) 1998, p.161-197.

[Swartout et al. 1996] **Swartout, B., Patil, R., Knight, K. and Russ, T.**, "Toward Distributed Use of Large-Scale Ontologies," *Proceedings of the 10th Knowledge Acquisition, Modeling and Management Workshop (KAW'96), Banff, Canada*, 1996.

[Thibault 2000] **Thibault, P. J.**, "The multimodal transcription of a television advertisement: theory and practice," in A. Baldry (Ed.), *Multimodality and Multimediality in the Distance Learning Age*, 2000, p.311-85.

[Tsichritzis and Lochovsky 1982] **Tsichritzis, D. C. and Lochovsky, F. H.**, "Semantic Network Data Models," *Data Models*, 1982, p.210-224.

[Wactlar et al. 1996] **Wactlar, H. D., Kanade, T., Smith, M. A. and Stevens, S. M.**, "Intelligent Access to Digital Video: Informedia Project," *IEEE Computer*, **29** (5) 1996, p.46-52.

[Weber and Poon 1994] **Weber, K. and Poon, A.**, "Marquee: A Tool For Real-Time Video Logging," *Human factors in computing systems: "celebrating interdependence"*, 1994, p.58-64.

[Weinstein 1998] **Weinstein, P.**, "Ontology-Based Metadata: Transforming the MARC Legacy," *Proceedings of the 3rd ACM International Conference on Digital Libraries*, 1998, p.254-263.

[Weiss et al. 1995] **Weiss, R., Duda, A. and Gifford, D. K.**, "Composition and Search with a Video Algebra," *IEEE MultiMedia*, **2** (1) 1995, p.12-25.

[Whittaker et al. 1999] **Whittaker, S., Hirschberg, J., Choi, J., Hindle, D., Pereira, F. C. N. and Singhal, A.**, "SCAN: Designing and Evaluating User Interfaces to Support Retrieval From Speech Archives," *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999, p.26-33.

[Wollen 1972] **Wollen, P.**, *Signs and Meaning in the Cinema*, Bloomington: Indiana University Press, 1972.

[World Wide Web Consortium 2001a] **World Wide Web Consortium**, "W3C XML Schema", http://www.w3.org/XML/Schema, 2001. (a)

[World Wide Web Consortium 2001b] **World Wide Web Consortium**, "Synchronized Multimedia Integration Language (SMIL 2.0)", http://www.w3.org/TR/smil20/, August 07 2001. (b)

[World Wide Web Consortium 2004a] **World Wide Web Consortium**, "Resource Description Framework (RDF)", http://www.w3.org/RDF/, March 2004. (a)

[World Wide Web Consortium 2004b] **World Wide Web Consortium**, "OWL Web Ontology Language Overview", http://www.w3.org/TR/2004/REC-owl-features-20040210/, February 10 2004. (b)

[Zhang et al. 1993] **Zhang, H., Kankanhalli, A. and Smoliar, S. W.**, "Automatic Partitioning of Full-Motion Video," *Multimedia Systems*, **1** (1) 1993, p.10-28.