A Framework for Modelling Video Content

Bryan Kinns, Nicholas Jonathan

Department of
Computer Science

Technical Report No. 759

QUEEN MARY
AND WESTFIELD COLLEGE
UNIVERSITY OF LONDON

ISSN 1369-1961

A Framework for Modelling Video Content

Nicholas Jonathan
Bryan-Kinns

August 1998

A Framework for Modelling Video Content

Nicholas Jonathan Bryan-Kinns

Submitted for the degree of Doctor of Philosophy

Queen Mary and Westfield College

University of London

August 1998

Abstract

Early developments in the capture and display of moving images concentrated on entertaining people. Through the years there has been a diversification in the use of such images. However, the complexity of moving images makes them difficult to process directly; external descriptions are needed which can be processed with respect to the moving images. Current approaches restrict the use, and reuse, of such descriptions and their associated images. This is because they are either too application specific, or too general.

This thesis posits that employing a semantic data modelling based framework will support the description of the content of moving images (models of video content in this thesis) *and* the description of structural and semantic regularities of these models. This will support the description of a cross section of models of video content which was not previously possible. Furthermore, such an approach will support different kinds of reuse of both video and its descriptions.

This thesis contributes a demonstration of the use of the framework to describe three markedly different real world models of video content. From this demonstration it is argued that the framework can describe other kinds of video content models. Furthermore, this thesis contributes an investigation of the feasibility of using the framework to support reuse, especially between applications which is currently poorly supported. In realising these contributions this thesis also contributes a survey of current work in the area, and suggestions about how such work could be developed in light of the thesis.

Acknowledgements

Thanks are due to my supervisors Peter Johnson and Sylvia Wilbur. Many thanks are extended to Adrianna Ratke and Fraser Hamilton for their patient help and personal support. Further thanks are due to members of the HCI group at Queen Mary & Westfield College who helped with various aspects of this thesis.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

This chapter provides the background to the thesis. The first section describes the development of technology to capture, display, and store moving images. Section 1.2 describes possible future directions of such technology and raises some problems that need to be addressed to realise such possibilities. Section 1.3 presents the goals of the thesis which is followed by a brief description of this thesis' contribution. Finally a summary of the structure of the thesis is given.

1.1 Background ('Waiting for the gift of sound and vision' [23])

The ability to capture and display moving images is one of the major technological achievements of the past 150 years. The basic principle is that a sequence of still images are captured in some way so that they can be presented to a viewer in rapid succession. When the rate at which the images are displayed is high enough (typically around 24 images per second) the viewer perceives a moving image rather than a sequence of still images. Conventionally this is achieved using a strip of celluloid film; a camera creates a sequence of still images on the film, once processed, the strip of film can be presented in some way. In the cinema such images are projected onto a screen.

1.1.1 Cinema

From cinema's humble beginnings in the 1890s the moving image has gripped the public's imagination. These early monochrome moving images seem primitive when compared to modern developments of the same techniques, but they delighted audiences of the time leading to a large growth in cinema in the early part of this century. Further developments have allowed the moving image to portray more aspects of the world around us. These include synchronised sound in the late 1920s and colour in the 1930s [46]. At this point viewers had both sound *and* vision.

1.1.2 Television

The BBC's first television service started in 1936 [29] and provided an alternative means of creating and displaying moving images for large audiences. Television relies on the same moving image principles as celluloid film: a sequence of still images is captured by a camera, this sequence of still images is presented to viewers to create the illusion of a moving image. Unlike celluloid film, television is an electronic medium. The television camera produces an electronic (video) signal which encodes a sequence of still images. Originally television did not involve any storage of the signal; images were captured by television cameras and transmitted to television sets which displayed them. Furthermore, as with early film, early television was monochrome (though unlike early film it included sound). The development of storage media such as video tape in the late 1950s and the development of colour in the late 1960s allowed television to compete with cinema by allowing full colour productions to be recorded for transmission at later dates. Such developments spurred competitive developments in cinema such as the production of better sound systems and larger projection screens [21]. To this day cinema still provides a higher fidelity of moving image and sound than can be achieved with television. Though current and future advances in digital television [181] seem set to redress the balance by providing greater television picture and sound definition.

1.1.2a Its not all entertaining

Most moving image productions are aimed at entertaining people and are produced by large companies such as the BBC for mass entertainment. But mass entertainment is not the only use of moving images. Moving images can be educational, instructional, enlightening, and informative. For instance, news broadcasts and documentaries aim to inform a mass audience rather than solely entertain them. Moreover, moving images can be captured for historical purposes, later analysis (for instance in research), surveillance, or for government security. Moving images need not be intended for a large audience, nor necessarily be costly to capture; individuals create their own moving images in the form of *home movies*. These usually have a very small audience and tend to be less well produced than other moving image uses. This growing diversity in the use of moving images reflects the greater accessibility of technology to capture, store, and present them *e.g.* inexpensive video cameras intended for home use.

In short, the moving image's unique ability to capture events lends it to many different uses, not just mass entertainment.

1.2 The Future of Moving Images ('Holding on for tomorrow' [5])

Conventionally moving images have been stored in a linear sequence on a linear storage medium such as video cassette or celluloid film. For creative purposes this means that the creator must decide on the sequence of images before they are stored. For example, a Hollywood film director uses the script and their imagination to decide on the final sequence of images that make up a narrative film. Once this sequence has been decided and the images stored it is very difficult to change the order of images as the medium determines their linear sequence. People who watch such linear sequences are passive viewers. They have no direct control over the sequence of moving images they are watching; they can change the presentation of the images, for example, stop the playback, or change the speed of playback, but the underlying linear order of the images is always the same. This is satisfactory for many purposes such as the conventional notion of narrative films in which the plot is decided before hand.

However, creating moving images using a linear storage medium is difficult unless the still images are initially captured in the correct sequence. Typically moving images (such as narrative films) are sequenced in laborious editing processes. Moreover, the desire to further engage the viewer and allow them to actively make decisions about the sequences of images they view requires a different approach. Instead of storing still images on a linear medium (which itself imposes a linear order on the images) there needs to be some way of storing the images so that they can be accessed in any order; they need to be stored on a *non-linear* medium. The order in which still images are displayed is then determined by some external representation. Importantly, this external representation of possible sequences can itself be influenced by the viewer.

Current systems [14][50][52][53][121][173] which support such non-linear approaches are computer based and store images on some randomly accessible video storage device such as a video disc. Descriptions of the content and logical structure of the stored still images, and possible sequences of images, are referred to as *video content models* and are the main focus of this thesis. These models are more easily manipulated by computer than video itself; the combined use of non-linear video and models of its content provides a new form of data which is rich in audiovisual content as well as being easily processed by computers *via* the associated models [44]. Moreover, they can be used by systems to support users in interactively influencing the presentation of video where the viewer's influence on presentation spans a spectrum from very little (similar to passive viewing such as watching a feature film) through to extremely active (such as a video based arcade game).

1.2.1 Reusing images

Capturing and storing the still images needed for a moving image consumes time and storage space. For many uses of moving images it would be preferable to *reuse* previously captured still images *e.g.* news reports are often reused in different news programmes which avoids having to recapture the images for each programme. Reuse can save time compared to the time it takes to capture the images in the first place, and could save storage space if the reused images were *referenced* in some way. However, reuse is not currently well supported. Typically it takes some time to locate and get permission to reuse appropriate moving images. Furthermore, the use of conventional linear storage media means that reuse must be performed by *copying* the images into their new context (as the sequence is determined by the storage media). Copying images takes time and effort. Further, it requires the same amount of storage space as newly captured images, and often results in degradation of the images' quality in their new situation.

Computer based solutions involving video content models could be used to address the problem of reuse. First, copying can be avoided by storing images in a non-linear storage medium and having descriptions of sequences of images separate from the medium itself. Therefore images do not need to be copied, they are simply *referenced*. This also means that the quality of reused images is not degraded as it is with copying. Second, computer based support for image storage and models of video content means that more detailed indices of the images can be created and used. With the appropriate location tools this could reduce the amount of time invested in searching for images to reuse.

1.3 Goals of the Thesis

Although great advances have been made in the computer based video storage there are still descriptive problems in terms of models of video content. Typically each approach idiosyncratically stores video content models. This is sufficient for individual systems but means that support for handling models must be recreated each time a new system is developed. Furthermore it makes reuse of images and their descriptions difficult. To reuse video between two systems it typically needs to be copied onto an intermediate storage medium and then copied into the new system. For example, to reuse a news report from one computer based news producer in another involves copying the news report onto video cassette or transmitting it *via* fibre optic cable [14]. Any descriptions of the video in the original system are typically lost in this reuse process and have to be recreated in the new system.

Similarly, many different systems may be involved in the construction of a moving image. With different computer based systems this involves copying video onto intermediate storage media between systems. This can reduce the picture quality, increases storage costs, and tends to mean that descriptions of video content are lost.

The goal of this thesis is to:

- Define a generalised framework for modelling video content and to demonstrate its ability to describe the structures and semantics of video content models used by a range of systems dealing with video; this is motivated by the currently poor support for reuse.

The framework will essentially provide a way of describing and manipulating video where previously different computer based systems were needed. In doing so it will support reuse of images and descriptions of video content by reference, even between what would previously have been different systems. This reduces the time, effort, and storage space required to support reuse compared to previous systems.

The subsidiary goal of the thesis is to:

- Investigate the forms of reuse supported by the framework.

1.4 The Thesis' Contribution

This thesis contributes to the engineering of computer based artifacts, in particular computer systems which support video storage and presentation. The thesis' contribution in this research domain supports further research within the domain as well as in other domains not considered in this thesis such as HCI, film theory, and the psychology of moving images. In other words it provides a tried and tested framework for modelling video content which further research can use. Specifically, the thesis contributes the following:

- A demonstration that one framework can describe the structures and semantics of real world video content models and that these models can be usefully associated with the video itself.
- A demonstration that the framework can describe the structural and semantic regularities of video content models.
- An investigation into the feasibility of using the framework to support different kinds of video content related reuse.
- An assessment of the *pros* and *cons* of using such an approach and suggestions about how the problems could be overcome by future work.
- A survey of current work in the field of video content modelling from a wider perspective than previously undertaken.

This thesis's approach is novel in the field of computer based support for handling video. As it is novel it sets out to demonstrate the utility of the approach; further research can then build upon this work to evaluate its utility and explore its use in other research domains *e.g.* studies of how a viewer could interact with different forms of non-linear video. Furthermore, the investigation of the feasibility of using such a framework to support reuse (an important use of such a framework) sets out the kinds of reuse that should be supported and makes subjective, introspective assessments of the framework's support for them. This assessment lays the foundations for future demonstration and evaluation of the framework's support for reuse.

1.5 Organisation of the Thesis

The chapters of this thesis are organised as follows:

- **Chapter 2: Video Content Modelling**

This chapter discusses video content modelling. It starts by defining what this thesis regards as a video content model, how this fits into current research domains, and what applications there are for video. Two paradigms of video usage are discussed and problems with current approaches are drawn out. Finally characteristics of video content models are proposed from the work surveyed.

- **Chapter 3: The Problem, Solutions, the Thesis**

Following from Chapter 2 this chapter focuses on the main problem identified in Chapter 2: the current inability of systems to abstractly describe video content models. This chapter then surveys different approaches to the general problem of abstract descriptions of models. It focuses on three computer science approaches which address this problem: object-oriented approaches, frames based approaches, and semantic data modelling approaches. The use of these different approaches is then considered with respect to hypermedia approaches. The rationale behind considering hypermedia approaches is that video content modelling can be considered a subdomain of hypermedia research. Moreover, there are currently no such approaches in video content modelling. Finally a suitable approach is proposed and the thesis is defined.

- **Chapter 4: VCMF Definition**

  This chapter uses the approaches proposed in Chapter 3, and the characteristics proposed in Chapter 2 to define a framework for video content modelling: VCMF.

- **Chapter 5: Supporting VCMF's Use**

  Using the definition of VCMF in Chapter 4 this chapter discusses the construction of a system to support it's use. Such a system is required because video content models often contain hundreds of objects and relationships and so would be tedious to construct by hand. Also, constructing models by hand makes relating annotations to segments difficult and cumbersome.

- **Chapter 6: Examples of VCMF's Use**

  The system developed in Chapter 5 is used to construct three real world examples of video content models. The intention of this chapter is to show that these three models can be constructed, and to investigate VCMF's support for reuse.

- **Chapter 7: Reflections**

  This chapter draws together all of the claims and issues about VCMF raised in the previous chapters. Importantly it considers what can be said about the generality and applicability of VCMF from the demonstration, and reflects upon VCMF's support for reuse as investigated in Chapter 6.

- **Chapter 8: Conclusions ('Felt like the end, but it was just the start' [18])**

  This chapter discusses the contribution of this thesis, in particular, how the work and problems identified in Chapter 2 are changed by VCMF. It then reviews the issues discussed in Chapter 7 and suggests how these issues can be addressed by future work. It also considers the implications of VCMF on other work.

Chapter 2 Video Content Modelling

This chapter reviews research which use models of video content. Section 2.1 defines key terms in this thesis: *video*, *content*, and *model*, and defines the phrase *video content model*. Then Section 2.2 provides a simple example of a video content model which serves to exemplify the terminology. Section 2.4 highlights the different uses of video content models in current literature. Section 2.5, Section 2.6, and Section 2.7 review the literature by examining two paradigms of video usage: the conventional (film based) paradigm, and the new non linear paradigm. These are followed by two sections which examine the characteristics of models and reuse of video shown in the approaches discussed in Section 2.6 and Section 2.7. These characteristics are summarised in Section 2.8 and Section 2.9, and drawn on in the selection of examples in Chapter 6, and the requirements defined in Chapter 3. The final two sections identify the research problems in current literature and summarise the chapter.

2.1 Definition of the Phrase *Video Content Model*

This chapter surveys video content modelling literature, but first it addresses the question *what is a video content model?* This phrase was briefly introduced in the previous chapter but a full definition is difficult. This is not only because of the different definitions in the literature, but also because of the different definitions of each of the terms: *video*, *content*, and *model*. To resolve these ambiguities, and to provide a clear definition of the terminology used in this thesis, the individual terms are defined first, followed by the definition of the phrase *video content model*.

2.1.1 Definition of the Term *Video*

Video is an electronically stored, transmitted, and displayed temporal medium; a progression of still images suggest a moving image to the viewer [191]. These still images are referred to as video *frames* [186] (frames for short) and are uniquely indexed when stored. They are usually synchronised with audio to provide an audiovisual presentation [44]. Conventional European television displays 25 frames per second to the viewer to give the illusion of movement. Each frame in this system is made up of 625 horizontal lines [210].

Frames are generated by some device such as a video camera which produces an electronic video *signal* [210] in which the frames are encoded. The frames may be viewed *live* as they are produced (allowing for transmission delays) and/or *recorded* on a storage medium (*e.g.* a video cassette, a hard disc) for later viewing which is referred to as the *play back* of a video recording.

Video storage technology can be simply classified along two dimensions: how the video signal is stored, and how it is accessed for playback. Table 1 illustrates a simple taxonomy of video storage media based upon these two dimensions with examples of typical storage media.

		Access	
		Sequential	**Random**
Storage	**Analogue**	Video cassette *e.g.* VHS video cassettes [126].	Video disc [126].
	Digital	Digital video cassettes used for professional archival [14].	Hard discs, Redundant Arrays of Inexpensive Discs (RAID) [133], Digital Video Disc (DVD) [7].

Table 1: Simple video storage taxonomy with examples of typical storage media

Considering the storage of a video signal first, it can be stored in *analogue* or *digital* format. These formats determine how the signal is encoded in the storage medium. Digital storage means that the signal is stored as *discrete* values whereas analogue means that the values are *continuous* (not discrete) [210]. The two main benefits of digital storage are: it can easily be processed by (digital) computers, and copying the video does not result in picture degradation as analogue based copying does [181].

Second, the video can be accessed *sequentially* or *randomly*. This thesis considers sequential access to mean that the video frames are stored in a single linear sequence determined by the storage medium. Moreover, to move from one frame to another, all successive frames in the sequence must be passed. For example, to move from the start frame of a video cassette to the middle frame, the tape must be physically wound from one reel to another to locate the middle frame. This involves checking all the frames from the start of the video cassette until the middle frame is located. Conversely, this thesis considers random access to mean that to move from one frame to another in the storage medium it is not necessary to pass all the frames in between. Random access storage media do not impose a single linear sequence on the stored frames. For example, to move from the start frame of a video disc to the middle frame involves repositioning the disc's reader; it does not involve passing all the frames in between.

It is important to note that although a storage medium supports random access of video frames, a user may only be provided with sequential access. Many of the systems discussed in this chapter use random access storage devices, but provide users with sequential access to the video. Later sections discuss the disadvantages of only providing users with sequential access to video. The framework defined in this thesis supports users' random access of video.

This thesis is concerned with the use of random access recorded video (shortened to video) which requires descriptions of the video's content. Moreover, in keeping with other research such as Chua *et al.*'s Video Retrieval and Sequencing System [44], this thesis assumes that video refers to a compound of visual *and* audio images which cannot be separately manipulated.

Furthermore, this thesis is concerned with supporting the description of the content of video recordings in a structured and systematic manner (discussed later). This is a research topic which has not yet been tackled, but it is one which provides a solid basis for the research domain and for a diverse range of future research. Uses of live video such as video conferencing [144], their associated network and transmission protocols (*e.g.* MBONE [33]) and their problems [107], the problems of storing and presenting video [77][201] to the viewer [187], the problems of capturing [2] and processing [120][134] video are not considered in this thesis. The strong basis that this thesis provides for future research supports research in these domains where they involve the representation of stored video content.

As mentioned in the previous chapter, film is essentially video's predecessor. It differs from video as its frames are stored as photographic images on a reel of typically celluloid film [21] as opposed to the analogue or digital electronic storage used by video [44]. With video storage a displayed frame may not be directly represented in the storage media. For instance, using compression algorithms such as the MPEG family [68], some frames are only stored as changes to other frames. However, it is still the case that the frames are electronically stored, and although frames may need be to be reconstructed in some way with reference to other frames, one still image after another is electronically presented to the viewer to give the impression of a moving image.

This thesis regards a frame as the smallest meaningful unit of information in a video (in keeping with Davenport *et al.* [51]). This is because a frame is the smallest unit of information that can be temporally indexed in a video. At the other end of the scale this thesis refers to the largest logical unit of information as a video *document* [182], such as a feature film or television programme. A user typically defines which groups of frames are part of the video document and which are not, though there are automated techniques discussed later in this chapter. The notion of a video document is especially important when a video storage device may store large amounts of video. In such cases video documents are used to indicate which logical units exist. Note that a video document does not necessarily have to be created solely using video. For example, a video document might be a feature film which was originally recorded using celluloid film, but which is stored on video. Also note that a video document itself might be split over several storage devices *e.g.* a feature film may be stored digitally on several laser discs. Moreover, video documents may have frames in common *i.e.* frames may be reused in different documents. An essential problem is that different uses of video are concerned with different granularity of video content and so their notion of a video document may be different. For example, one use of video might regard a single advert as a video document whereas another might regard one whole day's worth of television programming as a video document. Essentially, a video document is the largest logical unit for a given use of video.

To summarise, this thesis considers video to be a medium which is stored electronically, can be randomly accessed and processed by computer, and is displayed as a series of still images referred to as frames; such video might be stored on a video disc. Each frame is uniquely indexed and is usually synchronised with associated audio when presented to the viewer. This thesis is not concerned with *how* the video is stored or accessed; it assumes that there is some mechanism for uniquely addressing frames and that these frames can be located and displayed in some way.

2.1.2 Definition of the Term *Content*

A video recording is made up of frames. Each frame is displayed as a pixel array which may contain thousands of pixels. Each of these pixels may have colour and luminance attributes. This thesis does not consider any of these to be the *content* of a video recording. It considers content to be the *semantic* content of the video. Zhang *et al.* [212] highlight this distinction as they separate the 'intrinsic features of the video data' from the 'semantic content'.

The content of a video recording is therefore an interpretation (there may be many different interpretations) of the meaning of the images presented to the viewer as the video is played back. For example, a description of the content of a piece of video may be that it is the introductory part of the video document, or that certain characters are present and are having a discussion.

This notion of content helps to resolve the ambiguous definition of the term *film* which can refer to the celluloid medium as well as a genre of recordings. In this thesis, film refers to the genre, not the film medium. For example, a film such as *Citizen Kane* is recorded onto a medium such as a video cassette or a celluloid film. The term film in this case refers to the content of the celluloid film, not the celluloid itself.

2.1.3 Definition of the Term *Model*

There are few explicit definitions of the term *model* in current literature. A simple definition of a model is that it is a representation of some thing or things [148]. From an engineering point of view it is a 'small-scaled abstract representation of the real world' [197]. This thesis takes its definition from the definitions of Long [125], Palmer [148], and Tse [197] as follows:

- A model represents some aspects of the entity represented [148].
- Representation involves reflecting some aspects of the represented entity in some fashion [148]. For example, a map is a two dimensional representation of the geographic aspects of an area.
- Not all aspects of the represented entity need to be modelled, and not all aspects of the model need to represent aspects of the entity [125]. The reduction in the aspects

represented is systematic and explicit [125]. Furthermore, all unnecessary aspects in the represented entity which are irrelevant to the purpose of the model are not represented in the model [197]. For example, a driver's map might not represent the public footpaths in an area whereas these would be important aspects to be represented by a hiker's map.

To summarise, this thesis defines a model as something which represents some aspects of an entity; the reduction in aspects modelled is systematic and explicit, and determined by the purpose of the model.

2.1.4 Definition of the Phrase *Video Content Model*

From the definition of the three terms: video, content, and model, this thesis defines the phrase *video content model* to refer to a model of the content of a video document (there may be more than one video content model per video document). That is, it is a representation of an interpretation of the meaning of the images presented to the viewer as the video document is played back. Several works concur with this definition [24][55][123][185][212], others use alternative phrases. For example, the following can all be regarded as video content models: Davenport *et al.*'s strata [51], Tonomura *et al.*'s structured video [195], Davis' Media Streams [55].

As discussed at the end of this chapter, aspects of the interpretation of video content which are represented in current video content models are:

- The logical structure of the content.
- Interpretations of the video content related to these logical structures. These interpretations can be at varying levels of abstraction.
- Pertinent relationships between parts of these logical structures.

The primary purpose of these models is to provide a form which is 'more manageable in terms of size, complexity, human understanding, and computer processability' [197] than video which is particularly difficult to process [44][101].

Bearing this primary purpose in mind, a list (in increasing order of importance) of the aspects of video content which need to be represented in a video content model is as follows:

- At least it should give some description of the content of the video document. For example, a summary of the events that are captured in the recording.

  This is a very impoverished representation of the content of a video document. It does not represent any of the structural aspects of the video document and only provides a very limited description of the whole document. However, it is typically the only form of video content model provided by conventional multimedia and hypermedia systems [27]. For example, Gibbs' multimedia framework [76], Gaines *et al.*'s open architecture for multimedia documents [67], Intervideo [105], and the Amsterdam hypermedia model [86][31] all consider a video document to be an atomic unit of information and typically refer to it using some sort of simple identifier. As such, these systems provide a simple description of the content of a video document. This is impoverished when compared with the depth of description discussed in the following points.

Some systems such as Teodosio's Salient Video Stills [191] and Panoramic Overviews [192] present the viewer with a representation of the physical space captured in a video document. However, this is not regarded as a video content model because it does not provide the viewer with an *interpretation* of the content, rather it provides a *reconstruction* of the content.

- Better, it should also divide the video document into *clips* and associate them with *annotations;* this provides some representation of the logical structure of the content of the video document. A clip is a portion of the video document regarded as a logical unit [186]. An annotation is a piece of data that describes the content of clip [186] *e.g.* a piece of text, a list of actions carried out by people recorded in the video, or a representative image. As discussed in later sections, annotations can be generated by computer based analysis or humans. Moreover, different uses of annotations impose different semantics, even for the same clips. Annotations provide separately stored descriptions of the content of clips which can be more easily processed than the video itself. For example, annotations containing keywords may be searched to locate relevant clips. Annotations may represent different types of information *e.g.* one type of annotation might describe the actors appearing in a clip, another might describe the scenery appearing in the clip. Furthermore, they may contain information structured in some way *e.g.* a list of actors appearing in the clip. This use of clips and annotations allows the description of the content of the video document to be at a finer level of granularity than the previous level. For example, a list detailing the consecutive parts of a video recording (the annotations) and the position at which they start and end in the video (the association between the annotations and the clips). However, clips cannot share frames and so may not temporally overlap each other [55]. This provides a restricted partitioning of the video content.
- Even better it should represent *segments* of video documents and their associated annotations. Like clips, segments are portions of a video regarded as logical units, but segments may share frames and so may temporally overlap each other [59]. Therefore they provide a more flexible and detailed representation of the logical structure of a video document [55]. For example, segments may be identified in which particular actors of interest appear. These segments may overlap as different actors may appear at the same time.
- Best of all, it should represent the *relationships* between annotations of segments [59] (or clips). This allows the model to represent not only the content of logical units of video, but also how these units relate to each other. The most commonly used relationships are hierarchic composition relationships. For example, news

programmes are typically hierarchically composed of title segments and news item segments. News item segments themselves are typically composed of segments of the news presenter and segments of outside broadcasts [186].

The proposed MPEG-7 standard defines an approach which can describe all the aspects listed here; it will 'specify a standardized description of various types of multimedia information' [100] (including video). This recent development reflects the progress that has been made in video content modelling. It has steadily developed from simple descriptions of clips to the current stage of defining an international standard for the description of content at various levels of abstraction. However, as discussed in the rest of this thesis, there is no support for defining the general structure and semantics of video content models.

In summary, a video content model (model for short) represents a particular interpretation of the semantics and structure of a video document and at best consists of the following:

- Definitions of logical units of video referred to as segments.
- Annotations (any kind of data) of these segments. Annotations are descriptions of segments which are not stored in the segments themselves, but are associated with them and can be processed in some way.
- Descriptions of relationships between these annotations.

Note that to constrain the problem of describing video content models this thesis concentrates on structure and semantics. It does not consider the functional content of a video document; this reflects the characteristics of models reviewed in later sections. However, the framework defined in this thesis provides a strong basis to support future research into describing the functional aspects of video content.

2.2 A Simple Example of a Video Content Model

Figure 1 is a simple example of a video content model loosely based on work by Swanberg *et al.* [186]. Their work is concerned with the automated analysis of recordings of news broadcasts. Their system automatically segments videos into segments representing shots (video from one camera), and attempts to annotate these segments with information about the kind of shot it is: title, weather, anchor person, or news reel shot. The results of these analyses can be interpreted as video content models.

The figure describes the structure and content of a particular news broadcast on television at 10:15pm. This news broadcast starts with a title shot, followed by a single news report about "London's sun", and ends with a weather shot. The news report itself starts with a shot of the anchorperson, George, who provides an introduction to a newsreel about the sun in London. The newsreel is composed of two shots: one of Big Ben and one of the sun. Finally, the news report is summarised by the anchorperson.

Each piece of text in the diagram is an annotation of a segment. There are eight types

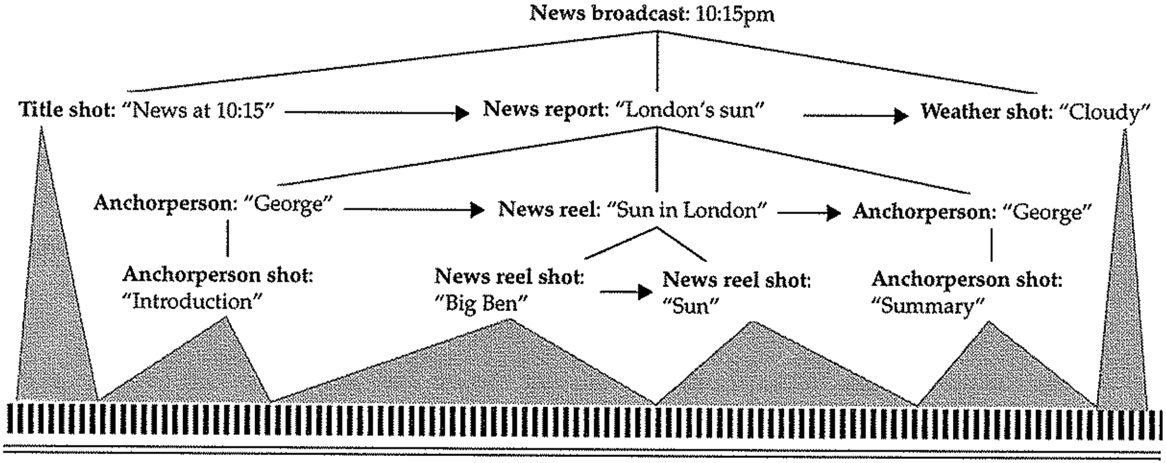

News broadcast: 10:15pm
Title shot: "News at 10:15"
News report: "London's sun"
Weather shot: "Cloudy"
Anchorperson: "George"
News reel: "Sun in London"
Anchorperson: "George"
Anchorperson shot: "Introduction"
News reel shot: "Big Ben"
News reel shot: "Sun"
Anchorperson shot: "Summary"

Figure 1: Simple video content model of a news programme

of annotation in this example (the type of a segment is shown by bold text): News broadcast, Title shot, News report, Weather shot, Anchor person, News reel, Anchor person shot, and News reel shot. In this example each annotation stores a simple textual description of the segment's content. The lines in the diagram represent relationships between annotations. Arrowed lines show sequence relationships and non-arrowed lines show composition relationships which work from the top to the bottom of the diagram. These are the two commonest forms of relationship used in video content modelling. Other forms of relationship, such as spatial relationships, are identified in the rest of this chapter. One advantage of the approach defined in this thesis is that it can describe a wide range of relationships rather than being restricted to these two as many other approaches are.

Note that segments are used instead of clips in this example because of the composition exhibited in this example; segments such as News report are composed of other segments and so temporally overlap them which is not permitted by clips.

At the lowest level of composition (indicated by grey triangles) segments representing shots are shown referring to actual frames of video which are represented by the stripes at the bottom of the diagram. This illustrates the association between annotations (annotations of shots in this case) and their segments (sequences of frames at the bottom of the figure).

2.3 Defining the General Structure and Semantics of a Model: Video Content Modelling Schemes

The previous example is a model of just one news programme. Many other news programmes exist which may be modelled in a similar way. The general structure and semantics of a video content model is defined by a *video content modelling scheme* (scheme for short). A model is realised from a scheme; it represents a particular interpretation of the content of a video recording whose structure and semantics are constrained by the scheme it is created from. As such, a scheme describes the structural and semantic regularities of a *class* of video content models (the set of models which are *realised* from the same scheme) *e.g.* the class of news programmes.

Figure 2 represents a scheme for the news broadcast example above. This is a very simple and somewhat artificial scheme, but it helps to illustrate the use of schemes and models. It states that, in general, such simple News broadcasts are sequentially composed of a Title shot followed by News reports which are followed by a Weather shot. The News reports are themselves composed of Anchor person and News reel sequences (in the example in Figure 1 the News report is concerned with the weather in London; different news reports will have different topics). These sequences are in turn composed of individual sequences of shots. This scheme could be used to provide the structure and basic semantics of all models of similar simple news broadcasts.

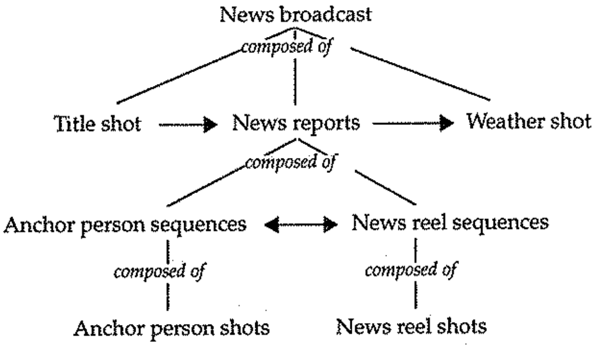


Figure 2: Simple video content modelling scheme of news programmes

2.4 Uses of Video Content Models

There are many more uses of video content models and schemes than the simple example illustrated in Figure 1 and Figure 2. The application of video content modelling is evident in domains where representations of the content of video recordings are used. There are several reviews of video usage and its domains which highlight the diverse uses of video [40][101][127]. These domains of use include:

- Interactive documentaries, learning environments, user interface evaluation, and multimedia communication [127].
- Surveillance and government intelligence agency applications [39][101].
- Training, simulation, medical applications, visual databases, and point-of-sales systems [44][101].
- Entertainment such as feature films, and information such as news [101].
- Educational applications [32][166][94][101].
- In the research domain video is used in tools which support the demonstration of research results and the analysis and exploration of behaviour. It is also used as an

archive for research findings and data as well as being a subject of research itself [40].

The domains discussed above can all be considered potential *applications* of video content modelling; they produce and/or use some form of model of video content. The applications themselves often do not regard their models as video content models, and usually have implicit modelling schemes, but they do store and use data about the content of video recordings. Later sections in this chapter interpret the use of such data in terms of video content modelling and infer the models and schemes required to support the applications. The characteristics of these inferred models and schemes are used to identify the requirements of a framework to support video content modelling.

The domains above are applications of video content modelling, but what research domain does video content modelling itself fit into?

First consider this thesis' notion of a video content model. It is regarded as a computer supported model which contains annotations, relationships between annotations, and associations between annotations and segments of video.

Second consider Chapter 3's discussion of the notion of *hypermedia*. In that chapter hypermedia is regarded as a loosely connected graph of nodes which can store various kinds of data. The arcs of the graph may have different semantics, but are generally referred to as *hyperlinks*.

From these two statements this thesis regards video content modelling as a subdomain of hypermedia research. The modelling is similar in nature, but different in purpose. Both video content modelling and hypermedia are concerned with graphs of information; segments associated with annotations which are related to each other in some way for video content modelling, and nodes of information linked to each other by hyperlinks for hypermedia. However, video content modelling differs in its purpose. It concentrates on *descriptive* models (of video content) whereas hypermedia models are primarily for *navigation* and *presentation* (of multimedia information). Video content modelling's descriptive emphasis is a result of the previously discussed difficulty in processing video itself. Models need to be constructed to support processing of interpretations of the content. These models may themselves be used to support navigation and presentation of video.

It is clear that video content models support video usage. Consequently, it is important to consider what structures and semantics are required of such models, and in what context they are used. Moreover, the intended application of video content modelling influences the structures[1] that will be present in the models [101]; different uses require different kinds of models. Rather than reviewing each piece of research with its schemes and models, this chapter considers how models are created and used throughout video usage. In particular, it considers the characteristics of models and

[1] The application also influences the functional characteristics, but as outlined previously, functional characteristics are not considered in this thesis.

schemes needed to support this usage. Essentially, this examination of video usage provides a framework for reviewing the characteristics of the schemes and models used and relates them to the processes involved in using video.

The next three sections provide a survey of research which creates and uses models of video content; regarded as video content modelling research.

2.5 The Conventional Paradigm of Video Usage

Video usage can be divided into two paradigms: the conventional paradigm of video usage which stems from film making techniques and produces a linear video document, and a new paradigm which produces non-linear video documents. The essential difference between these two paradigms is that there is one sequence of segments in a conventionally produced video document, and this is directly reflected in their sequence in the storage medium. For example, the shots of a narrative film are stored in the correct order in the celluloid storage medium. In contrast, there are many possible sequences of segments in a video document produced in the new paradigm; these possible sequences are determined by its model. The new paradigm typically requires some form of computer based support for sequencing segments for presentation.

This section and Section 2.6 concentrate on the conventional paradigm which most video content modelling research fits into; Section 2.7 discusses the new paradigm. Figure 3 characterises the conventional video usage paradigm; this is summarised in Table 2. The text indicates processes which are applied to video. The products of these processes are models and/or sequences of video segments. The arrowed lines indicate transfer of video from one process to another. This paradigm is based on the *studio* mode of making, storing, and viewing film because this paradigm has the longest history (dating back to 1920s [21]) and as the literature shows, it has influenced other uses of video [193]. The italic text in the figure refers to the phases of the studio mode of film making discussed in the following sub section.

2.5.1 The Conventional Paradigm's Historical Perspective

Bordwell *et al.* [21] suggest that the studio mode of film production is in some ways similar to that of the production line assembly of cars with one main proviso: although the studio mode of production uses a similar process each time (as does a car production line), it produces a different product at the end (unlike production assembly of cars which produces identical cars each time). They highlight major examples of the studio mode of film production as those created by Hollywood studios such as Paramount, or Columbia, between the 1920s and the 1960s.

In Bordwell *et al.*'s description of this mode of film production, the first phase is the *pre production* phase. In this phase scripts are written, and financial support is obtained. After this is completed, the *production* phase is entered. In this phase filming and sound recording takes place. Each shot (the smallest logical unit) of the script or storyboard (a

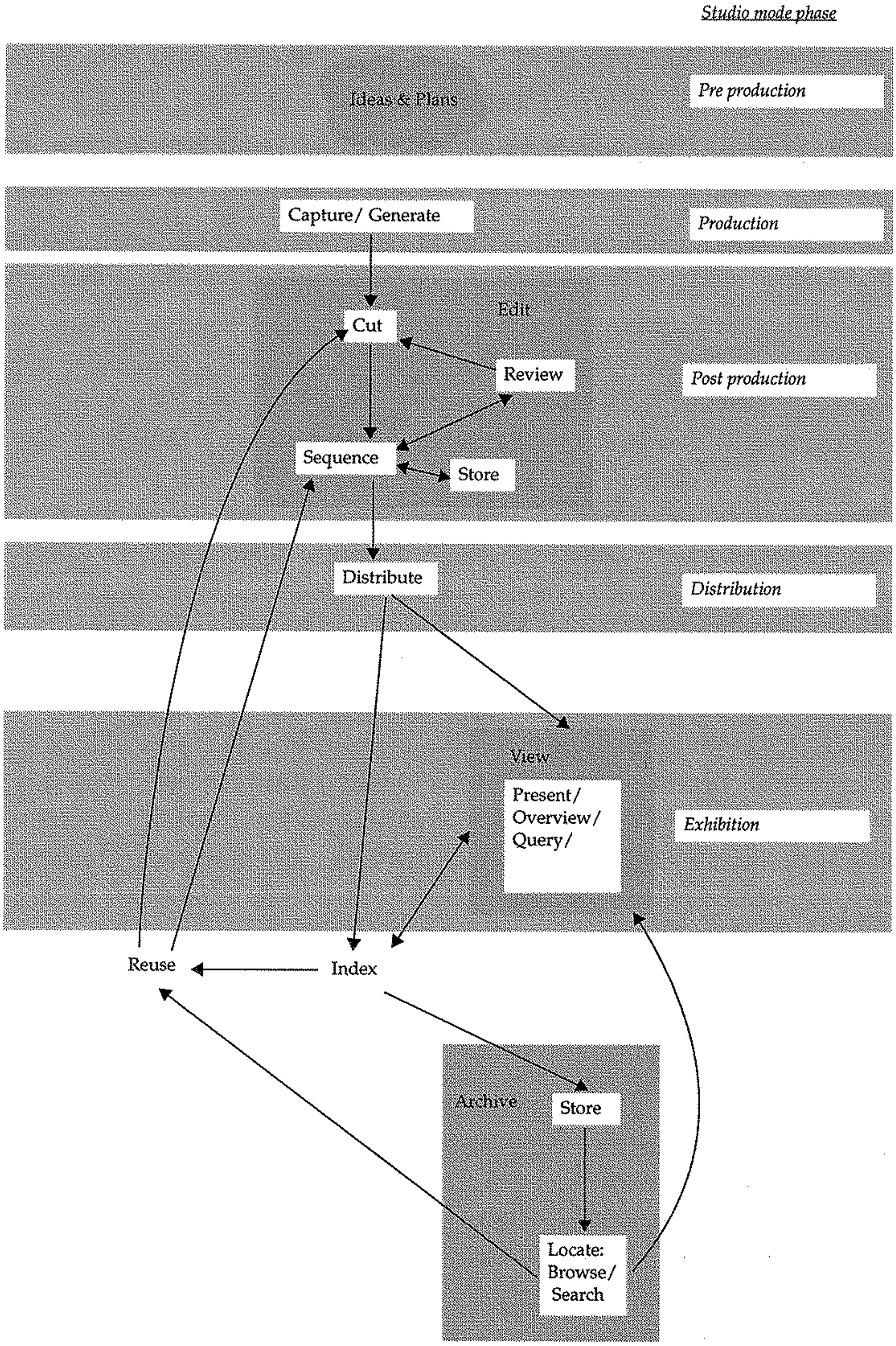


Figure 3: Characterisation of processes in the conventional video usage paradigm

visual representation of the script) is filmed in one or more *takes* (a single piece of film from one camera). Each take starts with a view of a *clapboard* which details the number of the shot (to relate it to the script), and the number of the take. This can then be used to uniquely identify each take. Whilst the film is in the production phase, some work begins in the *post production* phase. In this phase the best take for each shot is selected (therefore each shot is also a single piece of film from one camera), it is cut to its appropriate length, and sequences of shots are created. These sequences of shots may be iteratively reworked until a satisfactory sequence is completed. At this point sound is mixed and over dubbed to synchronise with the images. Once the film is complete, various photo processing techniques are applied to create a *release print*. Copies of the release print are then distributed to cinemas for exhibition.

Copies of the release print are also stored in archives such as the National Film and Television Archive. These can then be retrieved at a later date for exhibition, or even the reuse of some of its footage in another production.

2.5.2 Current Use of the Conventional Paradigm

Although this paradigm was developed to support the creation of films as early as the 1920s, these phases of production are still in use today (though not as rigidly defined as in the Hollywood studio system). Moreover, current video research such as Tonomura *et al.*'s work detailing a model of the video handling process in structured video computing [195], and Nack *et al.*'s discussion of the editing process [139] describe very similar paradigms. For many uses, it is logically the simplest way of creating and using video: the video captured, satisfactory sequences of video are created, and then it is distributed and viewed.

Even with the introduction of random access storage discussed earlier in this chapter, and broadcast systems such as Video On Demand [124] this paradigm is still the most commonly used. For example, digital cameras can be used in the production stage, digital editing suites such as those produced by Avid Technology [13] can be used in post production, and Video On Demand systems can be used to archive and exhibit the finished product. However, there is no integration between these digital systems. Cameras produce takes which are typically stored on (digital) video cassette. Editing suites then copy the takes, store and manipulate them on hard discs, and finally produce copies on video cassette. This cassette is then copied and distributed to various interested parties including Video On Demand systems which copy video onto their own storage media for later retrieval and playback.

The problem of the lack of integration between processes is highlighted by the amount of effort invested in researching the automated (re)segmenting of video documents prior to archival (discussed in Section 2.6.6). Re-segmentation of a video document essentially involves attempting to recreate the original model of the document before it was distributed. For example, many systems automatically detect shot boundaries which are then used to segment the document in to its original segments (shots) [49]. However, the success rate of such systems is typically not more than 90% [49]. The need to re-segment video documents arises from the fact that the sequence of segments in a video document produced by the conventional paradigm is determined by the storage medium itself *e.g.* a video cassette. Therefore there is no need for an explicit model of the video document and so it is usually not retained. Section 2.7 discusses the new paradigm which requires an explicit model of video documents in order to determine the possible sequences of segments and so starts to remove this lack of integration.

2.5.3 The Conventional Paradigm's Applicability Outside Film Making

As discussed previously, this paradigm is based on the studio mode of film making, storing, and viewing. However, it is not restricted to film making; it is applicable to other domains which use video in a conventional manner *i.e.* domains in which video is stored on some linearly accessed medium such as a video cassette.

Initially, some other domains may not appear to fit into such a paradigm. A pertinent example is the analysis of observational data. Some uses of video in the analysis of observational data (typically those following an ethnographic approach, *e.g.* Vortac's analysis of sequences of actions for air traffic controllers [202]) initially appear to follow the following process: hypotheses generation (which could be regarded as the Ideas & Plans process), observation and capture of behaviour using video (capture process), analysis of behaviour captured in the video to test or formulate new hypotheses (view process). This initially appears to not use the edit, reuse, or archive processes. However, on closer inspection of the use of video in such studies it is evident that these processes are used, though often not in the conventional sequence described previously.

Recent approaches to observational data analysis such as ESDA [62] (Exploratory Sequential Data Analysis) describe the requirements and implementation of tools to support observational data analysis using video (such as analyses described by Ritter *et al.* [163], Frolich *et al.* [64], Vortac *et al.* [202], and Olson *et al.* [147]). Sanderson *et al.*'s [170] description of the foundations of ESDA includes a discussion of typical ways in which video, and models of behaviour exhibited in the video, are used in analyses. In this detailed discussion it is clear that video is not only captured, indexed (annotated), and viewed, but it is also reused to create presentations of important results for distribution to other researchers. This involves the reuse, edit, and distribute processes. In addition, Chow [40] gives useful advice to researchers trying to create presentations of their work using video which highlights the reuse of video from analyses.

Another pertinent aspect of Sanderson *et al.*'s discussion of ESDA is the description of the way in which the behaviour captured in video is represented. They logically divide video into units called *chunks* (segments) which represent interesting items of behaviour. These segments are associated with *codes* or textual descriptions (annotations) which provide descriptions of the behaviour exhibited in them. Further, different kinds of relationships can be created between the annotations, such as 'segment a refers to segment b'. These characteristics are also evident in video based tools to support such analyses such as MacSHAPA [169][171][172] and VANNA [87][88]. In the paradigm described here the combination of chunks, annotations, and relationships are regarded as a video content model. These models are either created as the video is captured in the capture process, or as the video is iteratively indexed in the index process.

The medical use of video is another domain which initially appears not to fit into the conventional paradigm. Nardi *et al.* [142] describe the use of live video images of neurosurgery operations to allow observers to view operations from geographically distant locations. The primary use involves the capture, distribute, and view processes. However, the video is recorded for further use as an educational aid, and to support later analysis of operations themselves. This secondary use involves the archive (storing the recording), locate (finding appropriate recordings), and view processes. Furthermore it may use the edit process to create edited versions of operations which include pertinent parts.

2.6 The Conventional Paradigm's Processes

A general description of the processes (discussed fully in the following sub sections) used in this paradigm is given Table 2. To help illustrate these descriptions a simplified example of creating a documentary is considered.

It is important to note that different uses of video and models place different emphases on the processes in this diagram. For example, due to the large amounts of effort involved, creating a home movie often involves little fore-thought and planning, and usually involves little or no segmenting and sequencing [17]. Also, analysis of a film usually involves only the location of the film in an archive and analysis; the analyst does not usually create the film before analysing it.

Section 2.7 describes research which is attempting to change this paradigm and its processes primarily through the use of random (non-linear) accessible storage.

The following sub sections survey literature which use models of video content in the conventional paradigm. It examines how their models are created and used by the various processes characterised in Figure 3 and infers what schemes are needed to support such models. As discussed previously, this provides a framework for reviewing the characteristics of the models and schemes used. These characteristics are summarised in Section 2.8.

2.6.1 Formulating Ideas and Plans: The Ideas Process

The first stage of creating a video document is the formulation of ideas and plans. Ideas give the general purpose of the video document *e.g.* to create a documentary about sea mammals, to record all people entering or leaving a building, to record use of a

Process	Actions	
Ideas	Initial ideas and plans about the use of the video are formulated. For example, deciding to create a documentary and hiring a script writer to write its script.	
Capture	Video footage is captured using some device such as a video camera. For example, camera operators record narrative sequences for the documentary. The footage (video recordings in this thesis) may be stored on video cassettes which are then passed to editors who create the sequence of the documentary under the direction of the director.	
Edit	A video document is created from captured and/or reused video. Some uses of captured video, such as video analysis [128] involve no editing whatsoever as they are interested in the sequences of behaviour captured in the video. In such uses the edit process is not usually carried out. This edit process typically involves several iterations of segmentation and annotation of the video (cut process), creating sequences of video (sequence process), and reviewing these sequences (review process) until they are satisfactory. In film making editors and directors are involved in this iteration.	
	Cut	The footage is segmented (a subsequence of frames is selected; typically unwanted frames are disposed of) and annotated. In conventional film making there may be several takes for one shot. The most appropriate take for a shot is selected and cut (segmented) to a suitable size. This may involve copying the appropriate footage into an editing suite such as those produced by Avid Technology [13] where the segmentation and annotation takes place. In a conventional film making the annotations contain the shot, take, and scene number which relates it to the script or storyboard. The footage does not necessarily come from the capture process; it may come from the reuse process. In this case the footage is usually copied from the archive before being segmented if necessary.
	Sequence	The segments are put into some sequence. In the documentary example these segments are shots; their sequence is defined by the script.
	Review	The sequences are reviewed. If they are not satisfactory then they may be re-sequenced, the footage may be re-segmented, and additional footage may be captured or located. In addition, the original idea and plan may be altered by this review. Which may, in turn, affect the capture, segmenting, and sequencing of the video. This review may cause the edit process iteration to happen several times before the sequences are satisfactory.
Distribute	Once a satisfactory complete video document has been generated it is distributed by the distribute process. For example, a copy of the documentary may be distributed to a television company for broadcast, or copies may be distributed to video shops to be archived ready for customers. Note that this means that there are at least two copies of the video document at this point in the paradigm; one which is used in the edit process, and others which are distributed. The video document used in the edit process has a much more detailed model associated with it. This model may contain the shot, take, and scene numbers, as well as the script and/or storyboard itself. In contrast, distributed video documents typically have very little information associated with them. Some uses of video documents such as video analysis [128] do not use the edit process and so only one copy exists at this stage. Such uses typically do not distribute copies of the video document for public presentation; they are indexed and stored for analysis.	

Table 2: Summary of processes in the conventional paradigm

Process	Actions	
Index	Before storage (see store process), the video document is annotated in some way to support later indexed retrieval. If sequences within the video document are to be annotated then this process also involves segmentation. For example, the documentary may be indexed according to the topics of each scene. This involves annotating the documentary with information that is useful as a topic index. The segmentation involved in this process differs from that in the edit process. In this process the segmentation is used to mark out segments within a complete video document whereas in the edit process it is used to define segments which are then sequenced into a video document. A video librarian may be involved in creating suitable annotations.	
Archive	A video document is stored in some way for easy future retrieval. For example, the documentary may be stored in an archive such the National Film Archive (see [29] for a discussion of television programme archival), or stored in a video shop for customer selection. This process is composed of two sub-processes: store and locate.	
	Store	The video document and its model (created in the index process) are stored in some way so that they can be retrieved by the locate process. With storage media such as film and video cassettes the video documents are usually stored on physically separate devices *e.g.* on separate cassettes or reels of film. Conversely, with storage media such as large hard discs the video documents are stored in the same physical devices, but there is some database management system which allows the document to be located and played back.
	Locate	Video documents stored in an archive are located in some way, either through searching or browsing [165]. For example, a customer may browse a video shop, come upon the documentary and buy it. Alternatively, a researcher may search an archive for videos about a topic (that the documentary deals with) and locate the documentary. Located videos are typically presented by the view process or reused *via* the reuse process in other videos.
View	This involves a combination of presentation, overview, and querying of the video and its model. For example, the documentary may be presented to viewers by a television company, or the documentary may be queried to locate interesting shots.	
Reuse	Video documents located in archives or identified through analysis may be reused in the construction (through sequencing) of other videos. For example, the documentary may make use of footage from other video documents to illustrate certain points. These pieces of footage are located and then incorporated into the documentary. This usually involves copying the footage from the archive as the reused footage may need to be re-segmented to fit it into its new context.	

Table 2: Summary of processes in the conventional paradigm

system for later analysis. The plan defines the structure of the video document to be created *e.g.* the script of a documentary, or simply how long people will be recorded entering and leaving a building or recorded for later analysis.

In film making this is the pre production phase; story ideas are generated, storyboards are created, scripts are written, actors are hired, sets are built *etc.* The use of storyboards has been developed in the film industry to support the sketching out of ideas for the content and sequence of films. They typically consist of a sequence of sketches of the action and dialogue in each shot (a single piece of footage from one camera) of a scene (action taking place in one setting). They are used in conjunction with scripts in the filming (capture process) and editing (edit process) of a film.

Until recently these important products (the script and the storyboard) have not been related to the products of later processes. For example, storyboards are generally not directly related to the video document produced. This is not a problem when considering the finished product such as a video in a video shop, but it is a problem when creating the video. When creating the video it may be hard to: visualise the final result and communicate ideas about the final result during creation, rearrange storyboard elements and their associated video, or search for particular elements in the video based on the storyboard [129].

The approaches adopted by Mackay *et al.*'s Video Mosaic [129], Baecker *et al.*'s MAD system [15] (Multimedia Authoring and Design), and more recent professional editing systems [13] essentially try to link the storyboards and scripts to the video as it is captured and segmented. The scripts and storyboards in these approaches constitute the model of the video; they are static representations of the video. This model is easier to re-sequence and search than the video itself. Moreover, the linkage between the model and the video means that changes in the model, such as re sequencing storyboard elements, are immediately reflected in the video. This makes the capture, segment, sequence, review iteration easier as the video is sequenced directly from the storyboard or script, and can still be reviewed normally. Without this link, sequences of shots and scenes in storyboards and scripts have to be manually recreated in the video which can be time consuming, especially when several different sequences are tried.

Although the general approaches of Video Mosaic and MAD are the same, they differ in the finer details. Firstly, Video Mosaic uses *augmented reality* to support its interface whereas MAD uses a conventional desktop metaphor. Augmented reality in this situation means that computer generated and video images are projected onto a person's desk, and the viewer interacts with the system using a pointer or some other non-keyboard input device. Both approaches attempt to integrate the static medium of the script and storyboard with the temporal medium of the video. Neither system provides the ideal situation of a fully integrated display which is easily shared between interested parties, though Video Mosaic's use of paper representations does make the sharing of the static media easier.

Second, they differ in the relationships in their schemes. Both schemes use segments whose annotations are text with possibly other media such as drawings or audio annotation associated with them. But, MAD provides hierarchic composition relationships between annotations, whereas Video Mosaic provides association relationships between annotations. Ideally, both these kinds of relationships should be provided so that the storyboards and scripts can be hierarchically structured (reflecting the typical structure of narrative films), and parts of them could be associated with each other to provide simple non-hierarchic relationships.

2.6.2 Capturing Footage: The Capture Process

Once some ideas and plans have been formulated, video needs to be captured or generated. Capturing video requires the use of a video camera connected to some recording device; this captures a view of physical space in the video recording. Generated video is recorded onto video from a device such as a computer; it is not a recording of a physical space.

Most video at this point has little or no model associated with it. However, some research systems attempt to generate simple descriptions of video as it is recorded. These typically describe when and where the video was captured, and possibly some brief descriptions of segments. Davenport *et al.* [51] describe the requirements of a *data camera* which can store information about the camera position and movement, as well as voice annotations of who, what, and why information. Cruz *et al.*'s STREAMS system [47] uses multiple cameras to record meetings and lectures. Their systems manually meets some of Davenport *et al.*'s requirements as they encode the date and time of the recording as well as the camera id. Kazman *et al.*'s Jabber [108] takes this form of annotation one step further by proposing that speech recognition is used in real time to create textual annotations of video conferences as they are being recorded. However, their prototype system uses *post-hoc* manual annotation which reflects the difficulties involved in annotation during the capture process. Jabber also allows users to manually relate segments of video with agendas of meetings as meetings progress. However, this places a heavy burden on the person performing the annotation.

Video analysis systems such as EVA [128], DIVA [130], and DRUM [131] allow users to annotate the video as it is captured *i.e.* descriptions and segments are dynamically generated as the video is captured. In such systems the user typically presses keys on a keyboard to indicate that a particular (coded) annotation should be added to the model at that time. These coded annotations are then reviewed and refined at a later time (view process), typically to help test hypotheses about behaviour captured in the video. Gabbe *et al.* [66] and Cheyer *et al.* [39] constructed systems which attempted to automate this coding process. Triggers are defined, and when the video being captured matches the trigger (for example a change in the camera angle) a coded annotation is added to the model. These annotations are then used to help support later browsing of the video (overview process). In Gabbe *et al.*'s system each annotation contains an *IER* (Iconic Episode Representative) which represents the content of the segment; it may be a frame from the segment, or a user defined graphic. Similarly, Taniguchi *et al.*'s system [190] attempts to select representative frames (discussed in later sections) from video being captured in order to support viewers' browsing of the video at a later point.

All these systems attempt to provide some form of annotation of segments as the video is captured. These annotations are intended to be used as the basis of browsing or searching once the video has been captured. They do not, however, generate any relationships between the annotations; relationships such as composition must be created once the video has been captured. Also, the annotations tend to be crude (primitive and error prone) due to their constrained generation time. This means that the models generated are impoverished, often only containing clip definitions and simple annotations.

2.6.3 Constructing a Video Document: The Edit Process

This process takes video from the capture or reuse processes and iteratively segments and annotates (the cut sub-process), and sequences (the sequence sub-process) it until a satisfactory sequence is created (see Nack *et al.* [139] for a more detailed discussion of editing). Some uses of video documents may not use this process at all. For example, sociological data analysis in which the original ordering of events captured in the video is of importance, or home video creation in which there is often little inclination to expend the time required.

However, editing is an important part of many uses of video. In particular, film makers rely heavily on editing to create their films. Indeed, the effect of sequencing shots on a viewer's interpretation of the shots has been studied by many film theorists through the years [99].

2.6.3a Segmenting video: The cut process

This process segments and annotates recorded video which has been captured or located in an archive. As such, it differs from the capture process which attempts to segment video as it is recorded. This results in more time being available to segment and annotate the video in this process. Therefore more accurate and complex annotations can be created, and relationships can be defined between them.

This form of segmentation is also different to the segmentation that occurs in the index process. In this cut process the user is interested in defining segments and annotations which can then be sequenced to *create* a new video recording or presentation; frames are typically discarded. In the index process the user is interested in extracting information from a *previously* created video sequence, and using this to support indexing in an archive. Although work by Davenport *et al.* [52] and Chua *et al.* [44] initially appear to merge these two uses of segmentation they are not part of the cut and edit processes. This is because they interactively create temporary sequences of video for immediate presentation to the viewer containing the results of some sort of query of a video archive (see index, archive, and locate processes). They are not intended to be used to create sequences which are stored for later viewing as the edit process does.

In conventional film creation this process involves the physical cutting of filmed footage into shots [21]. These are then grouped into shots belonging to scenes, and are sequenced together in the next process. Modern solutions such as digital editing suites and the work discussed in the next paragraph often digitally store video on large randomly accessible storage devices. This allows *virtual* cutting of footage into segments; the footage is not actually cut, instead the editor defines the subset of the footage which is of interest.

Systems designed to support video document creation, such as MAD and Video Mosaic discussed previously, Csinger *et al.*'s IVAPS [48] (Intent-based Video Annotation and Presentation System), Tonomura's MediaBENCH [193], Duda *et al.*'s algebraic video [59], Nack *et al.*'s AUTEUR [139] (a system which attempts to automatically create humorous sequences), and Auguierre Smith *et al.*'s Stratosphere [12] (a database for temporal media), aim in part to tackle the problem of segmenting and annotating video in this process. The problem is to identify segments in the video and annotate them with some useful description and an identifier. The segmentation in this process is typically based on a person's personal choice; they select segments they think will meet their aim in creating the video sequence. Because of this there are no automated segmentation algorithms available to support this process. The annotation of the segments is dependent on the kind of video document being created.

Segments of video based on a script or storyboard (such as those described by Mackay *et al.* and Baecker *et al.*) are usually annotated with relevant parts of the script or storyboard, and some sort of identifier based on its position in the linear sequence. As mentioned previously, it is at this point that video creation systems such as MAD and Video Mosaic relate the script or storyboard to the actual video footage.

Other systems such as IVAPS, AUTEUR, algebraic video, and Stratosphere are not aimed at script or storyboard driven video document creation. In these systems the plot of the video document is decided at a later point, typically during the sequence process. Such approaches annotate interesting segments of video with some description of the content. In IVAPS and AUTEUR, these are topic descriptions which come from a domain knowledge base in order to try to encourage consistent use of descriptions (see the problems below). In Stratosphere, on the other hand, textual descriptions of where, who, what, when, why and how are used. Algebraic video supports free text annotation of segments. These descriptions themselves are then used in the next process to create video sequences for distribution.

Segmenting video in this process is fairly trivial, usually a case of identifying the start and end frame of a segment in some way. Creating a descriptive annotation is much harder. Script driven systems such as MAD easily relate parts of the script to the segments as an annotation. However, when the script is changed it may become inconsistent with the video it is associated with which may in turn cause problems when sequencing the video. This is a problem of maintaining consistency.

On the other hand, approaches which are not script driven, such as AUTEUR, IVAPS, algebraic video, MediaBENCH, and Stratosphere, associate arbitrary descriptions with their segments. These are typically not changed once set and so there are no consistency problems between the annotation and the segment. However, the main problem with using these descriptions is the *syntactic ambiguity* problem [48] in which descriptions in the annotations (such as topic names) may not be consistently used through annotations.

So, the main problem in this process is maintaining the consistency of annotations with the video and with each other. This is particularly problematic when annotations and/or segments are changed in the edit process' iteration.

Schemes used in this process typically define segments, annotations, and some relationships between these annotations. As mentioned previously, MAD and Video Mosaic provide hierarchic and association relationships, as does algebraic video and the MediaBENCH. IVAPS provides relationships such as *is_a* which are used to structure the knowledge base from which topic descriptions are selected as annotations. AUTEUR uses a hierarchic story structure of sequence, scene, action, and sub action. Furthermore its knowledge base supports a network structure of several types of semantic links (relationships in a model) between actions. AUTEUR currently makes the most extensive and detailed use of relationships between annotations. Stratosphere, on the other hand, provides no explicit relationships between annotations.

2.6.3b Constructing sequences of segments: The sequence process

Once some segments and annotations have been defined they are usually sequenced in some way prior to being reviewed in the review substage.

In film making this is usually done by an editor who manually splices pieces of film together, or uses some computer based solution [13] to join video stored on the computer together. These sequences are called *rushes* [21]. They are reviewed by the editor and possibly the director in the review process. Several different rushes may be created for one sequence to test out ideas.

The computer supported film making systems discussed previously (MAD and Video Mosaic) support similar forms of sequencing. In these systems the sequence of segments is defined by the sequence of the script or storyboard. So, the editor changes the script or storyboard to change the sequence of segments. MediaBENCH and algebraic video also support the construction of sequences of segments, but the segments themselves are sequenced rather than reflecting the sequence of their annotations.

System such as AUTEUR, IVAPS, and Stratosphere, discussed previously, and IDIC [168] and VAbstract [155] follow a different approach to this sequencing problem. Instead of supporting the editor's task of sequencing the segments, these systems create their own sequences. They use rule based approaches taken from AI research to create sequences from story structures. Simple examples of such story structures based on examples in these approaches are:

Convince: Describe, Conclude

Pre-emptive rescue: Negotiate, Breakdown, Threaten renewed violence, Pre-emotive rescue

The first structure states that to convince someone about something you must first describe it, then conclude. The second more complex structure states that the story for a pre-emptive rescue starts with negotiation which breaks down, renewed violence is threatened, and finally the pre-emptive rescue is carried out. Combinations of such structures are used by systems to select appropriately annotated segments and sequence them.

Rather than using explicit rules to construct sequences, a recent version of the movie content analysis project [118] (MoCA; developed from work including VAbstract [155]) uses simple heuristics to select appropriate segments from an archive. The system also has two implicit rules: first, they ensure that a film's titles are always present at the start of its trailer, and second they ensure that the last 20% of the film is never present to avoid ruining the suspense of the full film. In MoCA the user can specify the trailer's genre prior to creation; typically different genres of trailer suit different purposes. The selected genre influences the proportions of the kinds of segments (close-ups of actors, action, dialogue, or text) that should be included in the trailer. As there is no utilisation of story structures the sequence of segments in the trailer reflects the sequence of segments in the original film.

Similarly, Christel *et al.*'s *video skims* [42] use analyses of video content to construct sequences of segments which provide an abstraction of a larger video document (typically reducing the length by an order of magnitude). Interestingly they use both the audio and video aspects of video to select appropriate segments. Their studies suggest that using both the audio and video aspects can improve users comprehension of the abstraction.

As mentioned previously, it is important to be clear that these systems are part of the edit process as they produce sequences of video for viewing and storage as opposed to the temporary sequences produced by similar systems in the view process. For example, systems in the edit process may create trailers for films [118][155] (short summaries of films for distribution). These systems may need to locate appropriately annotated segments in archives to be used in these sequences. This is done *via* the reuse process. Similarly, AUTEUR [139] aims to construct humorous sequences from a collection of video footage in a video database. In the prototype AUTEUR system such footage is captured solely for AUTEUR's use rather than being reused from an archive.

Models in this process represent sequences of segments. These sequences are either defined by physical sequencing of the segments in the video, or by using the sequence of annotations in the model. Film editing uses the first method of sequencing; segments are physically connected to create the sequence. The rule based sequencing systems also produce a video in which the segments are physically in the correct sequence. This is typically done by copying the appropriate segments into a new video in the appropriate order. The computer supported film making systems described here (MAD, Video Mosaic, and Avid Technology's solutions [13]) and AUTEUR define the sequence using the model; the segments are not necessarily in this sequence in the video itself, but are displayed in the order defined by the model. This sequence is defined using some sort of sequencing relationships between annotations to define which segment should follow another.

From the point of view of flexibility, representing the sequence in the model is advantageous as it can be easily changed without affecting the video document itself. However, this involves more effort on the part of the system to play back the sequences as they are not necessarily in sequence in the video document. The rule based sequencing systems (except AUTEUR) and the conventional film making approach make play back much simpler as the segments are already in the correct order in the video.

However, to pass the video document to the distribute process for further distribution it needs to be stored on a linear medium *e.g.* a video cassette. This is a constraint of the conventional paradigm which is usually satisfied by copying the video document onto a suitable intermediate medium such as a video cassette. The original video document (and its model) may be stored locally by the store sub-process to allow ease of access for further editing.

2.6.3c Reviewing the sequences: The review process

This process supports viewing of the sequences to determine whether they are satisfactory for their intended purpose. If they are not, the sequences may be re-sequenced in the sequence process, segments may be re-segmented in the cut process, or additional footage may be captured, generated, or reused. If the sequence is satisfactory then the video document passes to the distribution process. It is worth noting that some sequences from the sequence process, for example, those produced by the rule based sequencing systems are intended to bypass this process and go straight to distribution.

In film making, the rushes are viewed by the editor and possibly the director to determine whether they fit the script [21].

The computer supported film making systems support the review of sequences by presenting video segments in the sequences defined by the script or storyboard (*i.e.* the model).

The models used in this process are used to help presentation of the sequences, and to aid identification of segments that may need to be reordered or re-segmented. There is no amendment to the models generated in the previous process.

2.6.4 Summary of Production Processes

The processes considered so far are concerned with the *production* of a video document. These processes are: ideas, capture, and edit (composed of cut, sequence, and review sub processes). Systems such as MAD [15] and Video Mosaic [129] provide integrated approaches which support most of these processes. In general, the models used in these processes consist of segments of video which are related to annotations. These annotations may define the sequence of segments using sequence relationships (*e.g.* MAD and Video Mosaic), and may be hierarchically related to other annotations to represent the logical structure of the completed video document.

Baecker *et al.* [15] discuss several examples of the use of MAD to construct video documents. One of their examples concerns the construction of a film to demonstrate a collaborative writing system. This example is presented here to highlight how MAD and similar systems are used within the conventional paradigm:

- First the outline of the film is sketched out (the ideas process).
- Shots for the first few sections of the film are then proposed, and suitable narration developed (again, the ideas process).
- Video is captured according to the directions and requirements outlined in the script (the capture process).
- Capture video is inserted into the MAD system to allow previews of the film as it is developed. This involves the cut process to trim the video to the correct length, the sequence process if the script is reordered, and the review process to preview the film.
- Once a final version had been decided, the segments are sequenced using a conventional editing suite to produce a linear video document for distribution.

Baecker *et al.* claim that their MAD system allowed them to 'very efficiently... develop and refine a concept for a movie, write and edit the script, revise the script after hearing how it sounded and flowed, and preview likely video sequences for inclusion in the film' [15]. As with other systems in the conventional paradigm, the result of this production processes is a linear video document which is passed to the distribute process.

2.6.5 Distributing The Completed Video Document: The Distribute Process

This process involves the distribution of completed video documents.

In film making this involves distribution companies such as Odeon physically distributing copies of reels of celluloid film for exhibition. Newer approaches to distribution include the distribution of video from one newsroom to another using dedicated networks [14].

The systems described so far distribute various kinds of information to other interested parties. The rule based systems distribute completed sequences to viewers, for example, video tapes to be displayed to members of the marketing divisions of a company. The computer supported film making systems distribute finished video documents and may also make available the models used in creating the document (scripts and/or storyboards). These models are either linked to the video, or separated from the video *e.g.* paper based storyboards.

A major problem with this distribution process is that copies of the video document are made and distributed to various parties, but the models of the video document are often not distributed with these copies. Even if they are, later processes tend not be able to interpret them. This leads to the index process typically having to recreate these models based on the video alone which may lead to misinterpretations of the original definitions of segments and their meanings.

The video document (usually without any model) is either passed to the index process prior to the archive process, or passed directly to the view process. The public transmission of video, such as the transmission of television programmes, or video on demand [124] are not considered by this process. Transmission usually takes place from an archive by locating appropriate video and transmitting it.

2.6.6 Creating Descriptive Annotations to Support Retrieval: The Index Process

Video documents from the distribution process are annotated before being stored or viewed. These annotations are used at a later date to support indexing and searching of the video document. There are two levels of annotation:

1. Annotation of the *whole* video document. For example, the title of the video, its bibliography.
2. Annotation *within* the video document. For example, the names of the shots in the video.

2.6.6a Annotation within the video document

Considering the second form of annotation first, the video document must be segmented, annotated, and relationships between annotations defined. Often this involves recreating models from the edit process that were lost during the distribution process. Note that the purpose of segmentation in the edit process is to define segments which will be used to create a new video document whereas in this process segmentation is used to identify segments within an already constructed video document. Also note that segmentation, annotation, and definition of relationships may either be manual or automatic.

- **Manual segmentation of a video document**

Manual segmentation of video [92][75] can be very time consuming and error prone [44]. The person(s) involved must define the start and end frames of possibly hundreds of segments in a video document. There is a growing body of research concerned with the automatic segmentation of video. These systems typically work on the assumption that the video to be segmented was created from a film perspective. That is, the lowest level of granularity of segments in the video will be a shot which is a single piece of video from one camera.

- **Automated segmentation of a video document from a film perspective**

Automatic segmentation systems segment the video document in a content free manner using image statistics [203]. They use algorithms to attempt to detect the boundary between two shots. The two shots may meet discretely at a cut, or may overlap each other in some way such as a dissolve where the two shots visually merge into each other. It is important to remember that these systems are recreating the information contained in models used in the edit process. These algorithms are typically used to define the segments of video content models used to support random access to film content *e.g.* Video On Demand systems [124]. There are several surveys of the algorithms used to detect shot boundaries (such as [44][49][85][190][213]). In particular, Dailianas *et al.* [49] provide an interesting comparison of the main automatic segmentation algorithms. The three most popular algorithms are *pair-wise pixel comparison, histogram comparison,* and *interval histogram comparison*:

- **Pair-wise pixel comparison** [212][38]. This algorithm compares two consecutive frames made up of a 2 dimensional matrix of pixels. If the change in a pixel's intensity between the two frames is above a certain threshold then it is considered to have changed. If the percentage of pixels changed from one frame to another is over a certain threshold then the two frames are regarded as being at a boundary. The problem with this algorithm is that camera movements, and large object movements can cause a false detection. Also, this algorithm is only suitable for detecting cuts, it is unsuitable for detecting dissolves and other special effects at the boundary.
- **Histogram comparison** [212][186][183][203][61][193][198]. As with the pair-wise algorithm, this algorithm compares two consecutive frames which are made up of a matrix of pixels. This algorithm is different because a shot boundary is considered to have taken place if the difference between the histograms of the frames is above a threshold. This reduces the problem of mistaking camera and object movement for boundaries as the histograms will typically not significantly change in these cases. However, in keeping with the previous algorithm, it is only capable of detecting cuts (though with greater accuracy, around 90%).
- **Interval histogram comparison** [185][195][44]. This algorithm attempts to detect shot boundaries where the shots overlap and are joined by a dissolve. Instead of comparing two consecutive frames, this algorithm compares the change in histogram values over several frames. Success rates for dissolves and cuts can be around 80%, so although they detect more complex shot boundaries, they are not as reliable as the histogram comparison algorithm is at detecting cuts.

Hampapur *et al.* [85], Zabih *et al.* [211], and Yeo *et al.* [207] provide comprehensive sets of boundary detection algorithms which also annotate the boundary with the kind of join it is *i.e.* whether it is a cut or a dissolve, and what kind of dissolve it is. Zabih *et al.*'s approach is notable because unlike other approaches they do not use intensity data *e.g.*

intensity histograms. Instead they detect visual *edges* in frames and look for new edges which appear far away from old edges; just after a segment boundary many new edges appear far away from old ones. Using this approach they hope to reduce the problem of mis-detection of boundaries due to significant motion within segments. Yeo *et al.*'s approach is also notable in its direct use of MPEG compression artifacts. They use *dc* images (spatially reduced frames used as part of the MPEG compression format) instead of full size frames which reduces the computational expense of the algorithms. This is similar to the approach of Meng *et al.* [134] who additionally use MPEG motion vectors to detect boundaries.

Several systems [44][85][195][212][213] use combinations of the algorithms discussed here, and possibly other algorithms such as audio based algorithms [155] to improve their boundary detection rates to over 90%. However, there is still a need for manual intervention to ensure that the video document is correctly segmented.

- **Automated segmentation from a non-film perspective**

One of the few attempts at automated segmentation which does not use the film paradigm is the work of Ho-Shing *et al.* [94]. Their interest is in creating an on-line educational system which contains multimedia lecture notes. Part of the multimedia notes of a lecture is a video recording of the lecture itself. This video is segmented so that each segment contains a recording of a slide from the lecture. Their automatic segmentation algorithm defines a segment boundary to occur when there is no slide being projected onto the screen. This is typically when the slide is being changed by the lecturer. Such segments form the basis of a model of a hierarchically structured model of the lecture which includes textual lecture notes as annotations (discussed later).

It is interesting to note that although this is not a film based segmentation system, the segmentation algorithms discussed previously are similar. As with this approach, they typically define the boundary of segments to be when there is a large change in the histogram of the image. As such, there may be similar problems with their approach. These include mis-detection due to large movement such as a person walking in front of the camera, camera movement, or changes in lighting.

2.6.6b Annotation within the video document

Once a segment, or a set of segments have been defined in the model they are annotated. As with segmentation, this can either be done manually, or automated in some way (maybe with manual adjustments).

These annotations attempt to describe the content of the segment and may contain: textual descriptions, categorisation, descriptions of objects and their movement, or representative frames.

- **Manual annotation within the video document**

As with manual segmentation, manual annotation involves a person entering the annotation themselves which can be time consuming and laborious [153]. Most manual annotation is concerned with:

- Generating descriptive text and/or keywords [213][92][165][182][123][93][44][75] (which are difficult to generate automatically [153]).
- The selection of representative frames which are intended to act as a static representation of the whole segment [165][44][75]. Typically an annotation of a segment contains a frame number which is considered to be the most representative of the segment.
- Descriptions of camera movement [75][55] (which can reliably be automatically generated, see later approaches).

- **Automated annotation within the video document**

Manual annotation is not only time consuming, but also prone to error due to possible inconsistencies, subjectiveness, and incomplete annotations [44]. To relieve this burden, and to try to improve consistency of annotation, many automated annotation approaches have been developed.

- **Automated selection of representative frames for annotation**

One of the most prevalent approaches to automated annotation is the automated selection of representative frames, for example, the approaches of Zhang *et al.* [213] and Wactlar *et al.* [203]. The problem is to select a frame which *is* representative of the whole segment. The representative frame's frame number is then usually included in the annotation of the segment to indicate the most representative frame [165]. The simplest form of representative frame selection simply selects an arbitrary frame from the segment (for instance the tenth frame) [11][198][190][38][154]. However, this may often not be the most representative frame from the segment [209]. The representativeness of a frame depends on the purpose of its representation.

Other systems attempt to address this problem using image processing techniques to select what they consider to be the most representative frame [183][213]. For example, Ho-Shing *et al.* [94] are interested in video based lecture notes. They select the frame with the least motion in it to be representative as it probably gives the best image of the slide being presented. Smoliar *et al.* [185] describe the selection of the *average* frame of a sequence to be representative. Either this frame's pixels are most similar to the pixels of a computed average frame, or its histogram is most similar to the average histogram. Selecting just one frame to be representative can cause problems in itself as one segment may have different interpretations which would require different representative frames. Bobick [20] surveys different approaches to representative frame selection, and discusses the problems of the *dynamics* (the kind of activity recorded) of a segment affecting the choice of the representative frame. In contrast, Aoki *et al.* [8] attempt to reduce the number of representative frames for a video document. This is achieved by selecting representative frames for each segment, but only displaying ones which are visually dissimilar.

- **Automated identification of representative colours for annotation**

  Another automated annotation technique based on frames is the extraction of the representative colour for a segment [155][195]. In this approach the two or three most frequent colours for each frame are selected. Then the most frequent two or three of these colours are selected and stored in the annotation to provide an indication of the most representative colours of the segment. These colours reflect the background colour in a long shot, or the object colours in a middle shot. For instance, a long shot of a desert (yellow in foreground, blue in the background) could be distinguished from a long shot of an ocean (blue in foreground and background). However, these comparisons can only be very coarse; a medium shot of fields being ploughed (*e.g.* green and brown in foreground, blue in the background) may have the same representative colours as a medium shot of a forest.

- **Automated camera and lens movement analysis for annotation**

  Several approaches to automated annotation create descriptions of the camera and lens movement: *panning* where the camera moves from left to right, *tilting* where the camera tilts upwards or downwards, and *zooming* where the lens is zoomed in or out. These descriptions are used to help in the automated recognition of a film's style [61], or to classify a segment [38] for later selection [213] and reuse [155][198]. They could also be useful for a film theorist's analysis of the use of camera and lens movement within a film [21].

  Three motion detection algorithms are used:

1. **Optical motion vector fields** [198][38][61]. The optical flow of the segment is calculated using algorithms derived from fluid flow analysis. When the camera pans, small regions of frames move horizontally in parallel. Similarly, when it tilts, small regions move vertically in parallel. When the lens zooms in or out, small regions move radially. Analysing the camera and lens movement relies on detecting these flows, but such flow analysis and detection is computationally expensive [195].
2. **X-ray images** [195]. To address the problem of computationally expensive fluid flow analysis used in the previous approach, Tonomura *et al.* describe an approach based on an *X-ray image* of video. The creation of an X-ray image is discussed by Tonomura *et al.*, but it is rather involved and so is omitted here for brevity. Suffice to say, their algorithm is less computationally expensive than optical flow analysis, but can be less reliable.
3. **MPEG artifacts** [134]. Meng *et al.* use the motion vectors from MPEG compressed video to estimate camera movement. This reduces the computation required to detect camera operations, but relies on the video being encoded using MPEG or another

compression format which works on a similar principle.

- **Automated object detection for annotation**

Automatic annotation of the camera and lens movement provides annotation which is *content free*. That is, the annotation describes the use of the artifact that produced the video (*e.g.* a camera), rather than the objects recorded in the video. Some approaches attempt to annotate video with general descriptions of objects and their movement within the video, *i.e.* they attempt to provide some description of the content.

The simplest object matching algorithms [198] require some input from a human. In the algorithms a human selects an object in a frame which they are interested in. The algorithm then checks other frames to see whether the colour combination of the selected object is present. If the colour combinations match, the segment containing the frame is annotated to say that it contains the selected object.

Other approaches [20][80][38][57][203][193] use algorithms derived from computer vision and flow analysis research. These algorithms require no manual input; they annotate segments with descriptions of the objects within them (*e.g.* size, shape, colour, and position [80]) and their movement within the frames. For example, a segment showing just the balls moving in a game of snooker (*e.g.* 2 red balls and 1 blue ball) would be annotated with a description stating that there were two red circles, and one blue circle in the segment. It would also be annotated with descriptions of their trajectory. These algorithms tend to be time consuming and computationally expensive. An interesting way to reduce this expense is to take advantage of the way in which the video itself is stored [57]. Storage formats such as MPEG [116] compress video by storing the differences between frames rather than whole frames. Motion analysis algorithms such as those developed by Meng *et al.* [134] take advantage of this as the information stored typically represents moving objects in the segment.

- **Automated classification of segments for annotation**

Some approaches use object matching algorithms to infer some semantics about the video content. In the news analysis domain, the spatial layout of frames in a segment is used to infer the kind of segment it is [186][212], and to annotate the segment with a categorisation. For instance, Figure 4 illustrates four spatial layouts of frames considered to be *anchor person* segments by Zhang *et al.* [212]. An anchor person segment of a news programme contains the news presenter (the anchor person) and some sort of logo or title. Any segment containing frames with one of these spatial layouts is annotated with the anchor person category. An alternative approach uses object matching to identify objects such as news broadcast logos [61]. In this case, an anchor person segment would be regarded as one which contained a small version of the news programme's logo. Both approaches may make mistakes in classification, either spatially or object based; currently there is little difference in reliability.

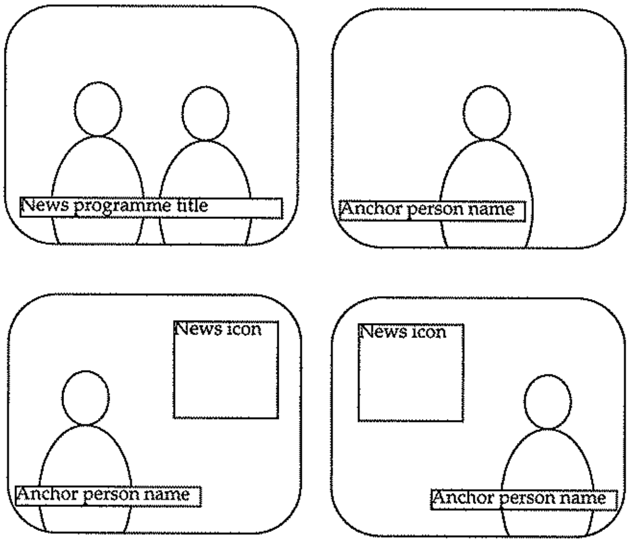


Figure 4: Spatial layout of frames in anchor person shots [212]

In order to help classify the segments, some approaches use simple finite state models to determine what the current and next segments could be. Swanberg *et al.*'s finite state model for news broadcasts [186] is illustrated in Figure 5. Their video parser starts at the first state and follows arcs when it detects a segment matching the arc description. For instance, after detecting the first anchor person segment it follows the news reel arc when it detects a segment regarded as a news reel. This reduces classification error rate as the number of choices is reduced (here to one or two choices at each state), but a model must be constructed for each kind of presentation which reduces the generality of the approach.

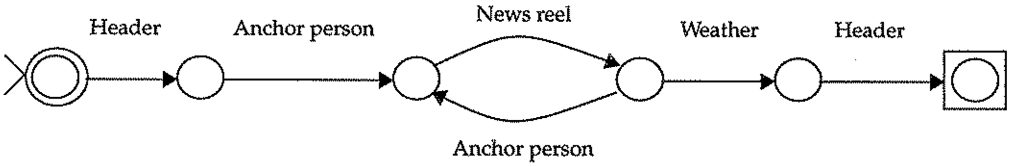


Figure 5: Finite state diagram of news broadcasts [186]

The MoCA project [118] has four categories of segment which it uses to create trailers for films: close-ups of actors, gunfire, explosions, and text. Close-ups of actors are identified using neural nets to learn the visual characteristics of faces whereas text recognition is supported by optical character recognition techniques. Interestingly they analyse the audio content to identify gun fire and explosions. This use of audio is uncommon in other research but highlights the depth of information available for analysis.

Kazman *et al.*'s Jabber [108] is another approach which uses audio characteristics to infer some classification of segments. They consider the relative temporal patterns of speech between two people and use these relationships to infer the following classifications: long uninterrupted oration, lively discussion, argument, brainstorming, and question-and-answer sessions. For example, a discussion may involve relatively even amounts of speech from both parties whereas a presentation involves one person speaking predominantly. It is not clear how well this approach would scale with more than two participants. Moreover, speaker identification may be problematic (though in their system each participant had their own audio stream recorded from a dedicated microphone which avoids this problem).

A further approach to segment classification is based on the temporal characteristics of segments and their neighbours. Taniguchi *et al.* [190] suggest that commercial films (advertisements) have the following properties:

"1. The length of a commercial film is 15 seconds or 30 seconds.

2. Commercial films often have more frequent scene changes compared to the program part.

3. A series of commercial film often occupies more than 1 minute." [190]

From this definition they aim to differentiate between commercials and programmes in television broadcasts, and annotate the segments appropriately. The main problem with this approach is its domain specialisation. This definition is only applicable to Japanese broadcast programming [190]. It may not be applicable to current British broadcast programming, and furthermore, may not be applicable to Japanese broadcast programming in the future.

- **Automated generation of textual annotations**

The problem of automatically generating textual descriptions of segments is imaginatively tackled by Brown *et al.*'s MEDUSA project [24], and Shahraray *et al.*'s system [179]. These projects extract the *teletext* transcript of a television programme. Teletext is an information service which transmits pages of information as part of the commercial television signal (not the picture) [210]. A teletext transcript is typically displayed as subtitles on a television set. These systems divide this textual transcription according to the automatically detected segments in the video, and use the transcript to textually annotate segments in the model. Shahraray *et al.*'s system additionally uses natural language processing to try to extract keywords from these transcriptions. Other systems use speech recognition to generate the same kind of transcript based annotations [41][42][203][183]. Although these systems do not rely on the presence of a teletext transcription, speech recognition is error prone and can be computationally expensive. A disadvantage of these various approaches is that the transcript typically only contains the words spoken in the video. Unless spoken, there will be no description of the action, characters, scenery *etc.* appearing in the video.

Another approach to automatic generation of textual annotations is the extraction of text from the video itself [94][117][203]. These approaches typically identify regions of text in video frames and then use optical character recognition algorithms to extract the text from the still image. The extracted text is then used as the annotation of the frame's (automatically detected) segment. Such systems are most useful when text is frequently used to describe the topic of a segment, for instance a news broadcast may use text to give a short description of news items. However, its use is limited as many video documents, such as films, do not contain any text (films typically contain text in their titles and credits, but annotation typically takes place in the narrative part of a film).

- **Modelling relationships within the video document**

Once a model contains segments which have been annotated in some way, relationships may be created between them. As with segmenting and annotating, the definition of relationships can be either manual or automated. In the literature reviewed here, the most common relationships used in this process are hierarchic relationships.

- **Manual relationship modelling**

The definition of relationships between annotations is a difficult problem to solve automatically. Most systems dealing with relationships provide support for manual definition of hierarchic relationships between annotations [92][165][123][93][44][38][193] [75]. These relationships are usually used to structure the annotations into some sort of film based hierarchy in which films contain scenes which contain shots (the atomic unit). Tonomura [193] additionally supports the definition of *reference* relationships which link two annotations if they reference each other. Similarly, Geißler's approach to hypermovies [75] supports the definition of hyperlinks between segments. These are links with no semantic meaning other than the fact that they lead from one segment to another.

- **Automated relationship modelling**

The automatic definition of relationships is complex as they are not easily inferred from the video itself. Tonomura *et al.* [198] provide simple links between segments which contain the same object. This allows viewers to easily locate segments in which the object appears, but otherwise provide little useful information.

Smith *et al.*'s system [186] attempts to automatically create hierarchic relationships between annotations. These group together segments that are considered logically and sequentially connected; these groups are themselves annotated with descriptions. Such groupings provide abstractions of sequences of segments and so may be useful when browsing the video document (see later processes).

Aoki *et al.*'s [8] and Yeung *et al.*'s [209] similar approaches cluster segments according to their visual characteristics and temporal locality. Each cluster is then assigned a label which is used to infer story units (similar to scenes in the film metaphor); a story unit has a minimum length and contains identical or recurring labels. For example, the sequence of segments assigned labels A, B, A, C, D, F, C, G, D, F is split into two story units: A, B, A and C, D, F, C, G, D, F. Yeung *et al.* then classify these story units into dialogue, action, or other. This classification is determined by the visual interplay of the segments: dialogue story units contain intense interaction between two dominant parties, whereas action story units contain segments with contrasting visual contents to express a sense of fast movement.

Yeung *et al.*'s classification rules are derived from film making techniques, whereas Aoki *et al.*'s are derived from studies of the visual characteristics of video documents. These approaches share a weak point in the initial labelling of segments. If this labelling does not accurately reflect the similarity of segments' semantics then the classification algorithms will not produce the desired results. In particular, only the visual dissimilarity between segments and their temporal locality is considered. Many other factors may affect the semantics of a segment including the audio and particularly the dialogue of the segment.

Lienhart *et al.*'s approach [118] does not use clustering of labelled segments. Instead they group segments together if they: share common colour content, have similar audio content, or if they can be identified as representing a dialogue. They do not automatically associate any descriptions to groups of segments; their groups are purely structural. Once again, the selection of heuristics which determine whether a grouping exists is crucial. It is interesting to note that the only visual characteristic used in their heuristics is colour. Considering other characteristics such as spatial layout [8] may help to improve the accuracy of their grouping heuristics.

Ho-Shing *et al.*'s system [94] relates segments of video recordings of lectures to parts of hierarchically structured lecture notes. Although the relation between the segments and the lecture notes is automatically defined, the content and the hierarchic structure of the lecture notes is manually defined beforehand. Moreover, the relation between the notes and the segments relies on the ability of the system to extract text from the segments which matches the lecture notes.

2.6.6c Annotating a whole video document

The annotation of a video document (as opposed to its constituent segments) is supported by a small number of systems. This may be because most approaches assume that the video is contained on some medium and is annotated outside their system. Manual approaches to description of a video document rely on text and keyword descriptions of the content [165][123][93]. For instance, the name of the film, its bibliography, genre, age rating.

Automatic annotation of a video document is even more scarce. Fischer *et al.* [61] describe their MoCA system which aims to automatically recognise a film's genre. They describe a three stage approach to film genre recognition which includes some of the techniques for annotating segments described earlier:

1. **Syntactic property analysis.** Including detection of cuts, objects, and object motions.
2. **Style attribute definition.** The style attributes are derived from syntactic properties. For example, scene lengths, and camera movement.
3. **Style profile identification.** The style attributes are compared to profiles of typical genres, and an educated guess is made as to which genre the video document belongs [61].

Once the third step is completed the video document is annotated with its probable genre. This genre annotation could be used to determine how the video is further analysed. For instance, video documents found to be news programmes could be further analysed using a system such as Swanberg *et al.*'s new analysis system [186] discussed previously.

In summary, the index process creates models of video documents which contain segments, annotations and possibly relationships between annotations. Furthermore, the annotations themselves are usually structured in some way, or they are typed to reflect the kind of information they contain. These are either created manually or automatically, though automatic creation tends to require manual intervention to correct errors. The models are used to support retrieval, either of segments or whole video documents. They often recreate some aspects of the models used in the edit process and lost during the distribute process. Although some of the approaches to annotation and segmentation here would be applicable to the edit process, their purpose is to create a model of a completed video document, not to help in the creation of the video document.

2.6.7 Archiving Video Documents For Future Retrieval: The Archive Process

This process deals with the storage, and retrieval of video documents. It has two sub-processes: store, and locate. Video documents enter this process from the index process, and leave to go to the display or reuse processes. These latter processes receive the video document after requesting it in some way from the locate sub-process.

2.6.7a Storing a video document: The store process

Conventionally video documents are stored on tapes in a library of some kind. For example the video archives of the BBC, and the National Film Archive, store video documents on video tapes in large archives indexed by the video's title (see [29] for a detailed discussion of British television archives). The video's bibliography is often stored with the tape, but indexes of the video itself are usually not usefully stored [55].

Digital technology is increasingly being used to store video to meet the growing desire to provide *Video On Demand* (VOD) services [124], and better indexing of video documents. A digital archive of films may need many gigabytes of storage. Even using compression such as MPEG, 72 hours (around 36 feature films) of full resolution, VHS quality compressed video would require around 64 gigabytes of storage space [24][124].

Typical storage technology (discussed in Section 2.1.1) includes: RAM, Hard disc, RAID [133], DVD [7], Optical discs, Magnetic tape; each technology has its own cost/ benefit ratio [124]. This large size of digital video storage is now practical in terms of cost and size, and provides several advantages over conventional storage:

- Faster access.
- No degradation of the video; tapes tend to degrade over time.
- Integrated storage of video and its model(s); conventionally a video's model is stored separately, often on paper making it difficult to search and use.
- Easier to directly process using computer based technology. This supports the use of automated segmentation and annotation techniques.

However, digital storage of video is still an emerging technology. Huge amounts of video is still stored on tape (and film stored on celluloid film) and simply indexed. These large sized conventional archives can not currently be practically stored digitally [55].

2.6.7b Locating a video document in an archive: The locate process

This process handles the location and retrieval of video (for instance a feature film) in a video archive. This thesis is concerned with video content models; as such it does not focus on with how models are used, but rather on what characteristics are needed of models when they are used. So, this thesis is concerned with what characteristics are required of models when locating video, not the task of location itself. Descriptions of approaches to video location are given to identify the characteristics needed of models.

Two kinds of location are considered: *search* and *browse*. Conventionally, these location operations result in the retrieval of one or more complete videos from the archive, even if the operation was interested in a segment of the video; segments of videos must be searched and browsed on a per video basis. This highlights one of the main drawbacks of the conventional paradigm; the lack of integration between processes.

The introduction of digital storage and retrieval techniques facilitates true segment based location, and integrates the locate and view processes (discussed in Section 2.7). This section is interested in the conventional approach to video use, and so retains the distinction between the two processes. As a result, this process is concerned with the location of video documents in an archive. These video documents are then passed to the reuse or view processes for possible within video document browsing and searching.

- **Searching for a video document: The search process**

The most common form of searching in a video archive involves posing a query in some query language; the result of the search is a set of video documents which satisfy the query. Conventional video archives in which the video is stored on magnetic tapes rely on simple indexes of the videos [55]. These indexes typically contain titles of videos, date of creation, some sort of genre classification, rating, physical location in the archive, and possibly some topic, plot, or bibliographic descriptions. When videos of interest are located in the index, they must be physically located and retrieved from the archive.

- **Searching for a video document in a digital store**

VOD relies heavily on digital video archives. These systems provide viewers with the ability to locate and view video remotely, typically in their own home. Several systems [165][182][123] have tackled the problem of providing indexes of video documents to support VOD style queries. Typical queries include the location of all 'movies produced by Universal in the 1960's in which Elvis did not appear' [165] and locate all 'movies in which Meryl Streep appeared in 3 or fewer scenes' [165]. The result of such queries would typically be a list of video documents which satisfy the queries. Furthermore, the viewer is provided with the ability to view movies in the list.

There are two levels of description in such archives: a description of the whole video document (a form of *overview*), and a description of the content of the video document. Overviews of video documents usually need to be manually created and maintained because of the lack of automated annotation at this level of description (discussed in the index sub-process). As with conventional storage of video, these indexes usually contain descriptions of: title of video, bibliographic abstract, comments, date of entry into index, release date, organisation that produced the video, length of video, genre, list of cast members, director of video, and awards won [165].

Unlike conventional indexing approaches, digital video archives tend to extensively annotate the content of video documents. This is partly due to the availability of automated annotation which reduces the time and effort involved in creating indexes, and also the requirements of VOD queries discussed previously. As VOD is mainly used for viewing and locating entertainment videos such as feature films [123], the content of the video documents is typically structured cinematically into a hierarchy of document-part-scene-shot. The lowest structurally lowest level, *shots* are unbroken sequences of frames from one camera. *Scenes* are composed of shots; they focus on one piece of action in the video document. *Parts* are composed of scenes; they represent logical divisions in a video document. These different levels of structure have associated with them different forms of description. At the lowest level, the shot level, there are descriptions of the shot, its start and finish time, the time of day, type of shot *etc.* [165]. Often these descriptions (such as the type of shot, and its start and finish time) can be automatically generated by the index process. At the scene and part levels, there are descriptions of the scene or part, its start and finish time, the time of day, the action the scene or part focuses on, cast members and objects appearing, geographic location *etc.* [165]. These varying levels of description are used to support queries such as 'movies in which Meryl Streep appears in 3 or fewer scenes'. This kind of query would first locate video documents in which Meryl Streep appeared (using the cast list of the overview description). It would then search in this subset of video documents for ones in which Meryl Streep appeared in 3 or fewer scenes. Other queries might involve searching for video documents with particular locations, seasons *etc.*

Simonnet *et al.* [182] highlight the need for these different levels of index. They describe four types of query that may be posed (based on iconographic database research):

- **Thematic**: queries which are motivated by a search related to a broad theme. For example, searching for a news video document containing the topics of ecology and nuclear power.
- **Exploratory**: queries which occur when a user wants an overview of the database content, for example, searching for all news video documents.
- **Precise**: the user knows, to a large extent, which video documents are needed. For example, searching for a news video in which the location is Paris, the description of action includes G7 summit, and inauguration.
- **Connotative**: queries which concentrate on subjective features. For example, searching for a video with topics of ecology and nuclear power, but with a comment that it is generally happy.

Digital video archives typically store their indexes in conventional databases [165] such as POSTGRES [199]. Such indexes are stored textually, and are queried using a query language such as PSQL [199] which provides powerful querying constructs. When information about a video document is located in the database, the video document is retrieved (*e.g.* for playback) from a video database (which may be integrated with the index database) using references stored in the index database.

So, to support the different kinds of queries of video content which result in a set of video documents, indexes not only need overview descriptions of the video, but also descriptions of its content. In terms of video content modelling this means that these kinds of queries not only use annotations of the whole video document, but also annotations of segments within the model. Furthermore, segments and their annotations, are usually related to each other hierarchically.

- **Browsing an archive for video documents: The browse process**

There are many different interpretations of the term *browse*. This thesis adopts Zhang *et al.*'s definition [213] which considers *browsing* to be the 'informal perusal of content which may lack any specific [search] goal or focus'. This contrasts searching where specific requests for information are generated (as queries), and a set of results is obtained. Often the results of poorly expressed queries are examined by browsing [213].

Conventionally browsing takes place simply by looking through some sort of index of an archive for something of interest, or actually browsing the archive itself. For instance, someone wishing to rent a video from a video shop often browses the shelves of videos in the shop to make a selection. Their focus is to rent a video; they may not know which video, or even which genre of video they want. Searching for a video to rent, on the other hand, may involve the customer asking the shop assistant for the location of the particular video and then retrieving it from the shelf.

- **Browsing for video documents in a digital store**

As discussed in the search sub-process, VOD aims to provide viewers with access to stored video, usually for entertainment in their own home. Two forms of browsing are considered here: genre browsing and virtual shelf browsing.

Genre browsing allows the viewer to select a genre of films (possibly from a hierarchy of genres) whose videos are then browsed using virtual shelf browsing. Figure 6 illustrates part of the genre hierarchy available in Rowe *et al.*'s Video Database Browser [165]. In this figure, the user has navigated to the genre of short computer animation movies (indicated by the thick border on the box; the arrowed lines show the path taken). Video documents in this genre would then be represented in a virtual shelf for further browsing.

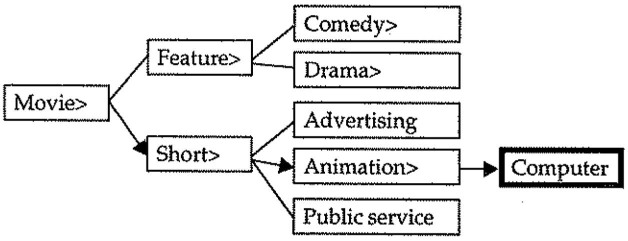


Figure 6: Part of Rowe *et al.*'s genre hierarchy [165]

Virtual shelf browsing presents users with a representation of the shelves of videos typically found in a video shop. Videos are represented by rectangles which contain the title of the film, and possibly some graphics associated with the film. Clicking on a rectangle causes the video to be retrieved and played. Figure 7 illustrates part of a possible virtual shelf of videos for the music genre (indicated by the music icon on each virtual video). The rationale behind this approach is that users will already be familiar with such a method of browsing and so will find it easier to use [123].

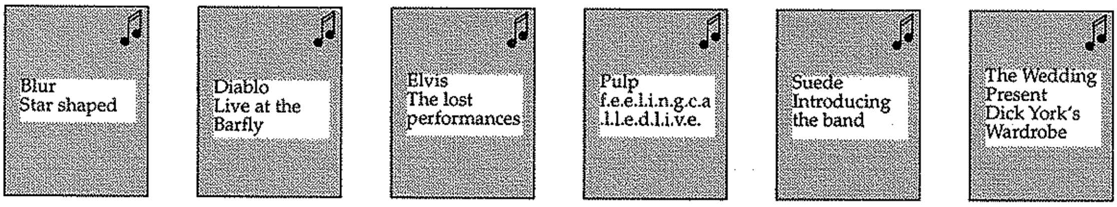


Figure 7: Virtual shelf of music videos

Browsing a VOD database for a video requires a video content model which annotates videos with their title, and their place in a genre hierarchy (note that one video might be a member of more than one genre). The model also needs to represent this genre hierarchy in some way, for instance by using annotations to represent genres and relating them to their sub genres using hierarchic relationships.

2.6.8 Viewing a Video Document: The View Process

As mentioned in the previous section, this thesis is concerned in the characteristics of models of video. It does not consider how such models are presented, but rather the characteristics needed to support such presentation. The previous section also discussed the conventional approach to video location and retrieval from an archive. This approach results in a set of video documents being retrieved for reuse or viewing. In keeping with this conventional approach, this process supports the viewing of video documents from the locate, index, or distribute processes.

Models used in this process (discussed in the following sub-processes) need to be able to represent segments, annotations, and relationships between these annotations [87][88]. The annotations themselves are either free text, structured in some way to provide attributes (text and value pairs), or contain a frame from the video. Further, it should be possible to logically group segments together *i.e.* use relationships between annotations to indicate grouping between segments. In the paradigm considered in Figure 3, these models would be generated in the index sub-process, but would not necessarily be archived. Often the video would be iteratively indexed and viewed; the viewing itself often influences the indexing as new annotations are created to reflect new items of interest discovered whilst viewing [87][88].

This process is subdivided into presentation, overview, and search sub-processes.

2.6.8a Presentation of a video document: The present process

Presentation displays a video document to the viewer. Conventionally the European system (PAL [126]) displays a frame as 625 horizontal lines which are refreshed 25 times every second. Recent high definition television display standards give a higher resolution *e.g.* 1125 lines demonstrated by the Japanese broadcasting company NHK in 1981 [21]. The aspect ratio of these displays is also different; 1.77:1 as opposed to the conventional 1.33:1. These displays are intended to provide an image which is clearer, and more like celluloid film images [181].

Additionally, annotations associated with the currently displayed segment may be presented. For example, EVA [128] (Experimental Video Annotation System), its development DIVA [130], and Geißler's Tour Planner and Guide [75] can synchronously display annotations of video segments as they are presented.

2.6.8b Providing an overview of video documents: The overview process

This sub-process provides the viewer with some sort of overview of the content of a single video document. Such overviews are used for browsing, searching, or previewing video documents. They typically provide a temporally reduced representation of the content of a video document with which an initial understanding of the content can be gained [208].

As defined in Section 2.6.7, browsing is the informal perusal of content which may lack any specific search goal or focus. Searching, on the other hand, has a defined goal: to locate a particular item within the video document. Previewing aims to provide the viewer with a rough idea of the content. Conventionally, linear devices such as video cassette players only provide simple linear controls such as fast forward and rewind. Searching, browsing, and previewing video are not easily accomplished with such controls. All the viewer is able to do is to increase the speed of the video's presentation (fast forward), and/or change the direction of the presentation (rewind). In either case they may miss information they could be interested in [195], or become confused about their orientation within the video document [137].

Some video overview approaches such as Tonomura *et al.*'s *flash browser* [195] display a sequence of representative frames (discussed in the index substage) of a video. Although this may not reduce a viewer's disorientation, it is claimed that the use of representative frames (which are intended to be representative of the segment) should reduce a viewer's chance of missing items of interest [195].

- **Spatial layout video's temporal dimension**

Typically the temporal dimension of video is represented spatially. For instance, the display of the temporal sequence of video frames as a spatial sequence. This provides the viewer with some idea of their orientation within the video document. If the information displayed is representative of the video content then it can also reduce the possibility of the viewer missing interesting information.

There are two related issues here: how can video be usefully spatially laid out, and, what information is representative. Most approaches spatially arrange individual video frames to represent the temporal dimension of video, and tend to assume that representative frames *are* representative. Whether or not a representative frame *is* representative is discussed in Section 2.6.6. This section considers how such frames can be spatially laid out.

As discussed previously, approaches such as Teodosio's Panoramic Overviews [192] present the viewer with a representation of the physical space captured in a video recording. This acts as an overview of the video, but does not utilise any model of the video and so is not considered by this thesis.

- **Spatial layout of video's temporal dimension: the video magnifier**

Simple versions of the Hierarchic video magnifier [137] do not use any model of the video document. They display a fixed number of frames selected at regular intervals from the video in a *story board*. Figure 8 illustrates the use of such a video magnifier. The frames in the story board are arranged from left to right in frame number order. The first level story board of the figure shows frames selected at regular intervals from the video document (here 1 frame per minute). The viewer can select different frames, or can *magnify* a frame. Magnifying a frame produces another story board below the original (the connection between the original frame and the magnified story board is shown in the diagram by the lines between the story boards). The frames in this additional story board are selected at regular intervals from the interval that the original frame was selected from (here 1 frame per 10 seconds). Hence, each story board represents a shorter time span in the video document. These frames themselves can be magnified. This process of magnification continues until there is nothing left to magnify, *i.e.* the story board contains frames which directly follow each other in the video. Although this approach could be regarded as creating a hierarchic model of the video document, it is not stored and so only temporarily models the video.

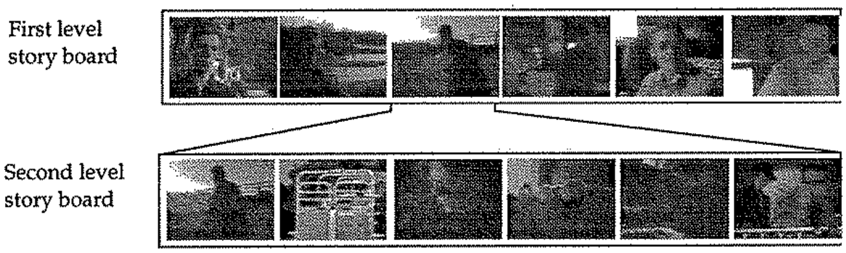


Figure 8: Video Magnifier [137]

- **Spatial layout of video's temporal dimension: the micon**

Simple versions of *micons* [198] (also called moving video icons [193], Video space icons [194], video streamers [47]) also make no use of models. A simple micon presents the frames of a video document as a three dimensional rectangle rather than the usual two dimensions. The third dimension represents time. Figure 9 illustrates an example micon. The figure shows the perspective presentation of the three dimensional rectangle. In this example there are 10 frames which represent 2 seconds of video playback. The edges of the frames are slightly exposed due to the perspective; this provides the viewer with some idea of the changes that occur in the video over time [47]. In the figure, the right hand edge of the micon shows some changes over time; this reflects the camera's downward movement. This technique can also be used to overview the movement within a frame if the micon's size is less than that of the frames of the video [47]. Additionally, the depth of the micon represents the temporal length of the video [198] so the viewer can determine the position of frames in the video from their position in the micon stack. Interaction supported includes the viewer *stroking* the edge of the micon which causes the frame the mouse is under to be displayed, or simple sequential playback [47].

- **Spatial layout of video's temporal dimension: laying out representative frames**

A common spatial layout which utilises a video's model is the spatial layout of representative frames [11][190][195][66]. In these approaches representative frames are

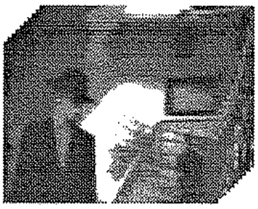


Figure 9: Example Micon

spatially laid out from left to right (and top to bottom if necessary) to provide an index of the video document. Because the representative frames are intended to represent the content of their segment it is claimed that they should provide a better overview than the arbitrary frame selection of the simple hierarchic video magnifier.

Arman *et al.* [11] extend the representative frame paradigm by scaling the presented representative frames to indicate how similar they are to a frame selected by the user. This is intended to aid searching as similar frames will be more visible than dissimilar frames, whilst the sequential relationships between representative frames are retained.

Yeung *et al.* [209] also use scaling of representative frames. Their approach is aimed at providing a brief static overview of a sequence of segments rather than reflecting the match of query results. Unlike Arman *et al.*'s approach [11], they use the size of the image to indicate the importance of that image in the sequence. The importance is either defined by a user in the model, or represents the frequency of similar images in the sequence.

Another extension of the representative frame paradigm is the additional presentation of some descriptive text from the model below frames [179][203][183]. This provides the viewer with two forms of representation of the video: text and still image. As such it provides more information in the overview than image or text alone.

- **Enhancing the video magnifier and micon**

Using a model's segments and representative frames in a hierarchic video magnifier provides the user with levels of magnification based on a description of the content rather than the number of frames. Zhang *et al.* [213], Yeo *et al.* [208], and Smoliar *et al.* [185] all describe the use of segments and representative frames in video magnifier systems. Instead of selecting frames at regular intervals from the video, Zhang *et al.*'s story boards are constructed from representative frames. Each representative frame comes from a group of shots (segments) in the video; magnifying the video reduces the number of shots until there is only one. Smoliar *et al.*'s approach has more in common with the original video magnifier approach. The video is arbitrarily divided into equally sized segments, but unlike the original approach, these segment boundaries are adjusted so that they are at shot boundaries.

Micons can similarly be enhanced. It is claimed that ensuring that the initially displayed frame of the micon is its representative frame, and displaying a textual description of the content of the video [212][198] gives the viewer a better idea of the content of the segment.

- **Spatial layout of video's temporal dimension: the time line**

Another spatially oriented paradigm for video overview is the *time line* [55][12][39] [89]. Again the temporal dimension of video is graphically represented from left to right on a screen. In contrast to representative frame based approaches, time lines concentrate on the temporal aspects of segments and their annotations. These are represented by graphic objects such as rectangles or lines; the left hand edge of a rectangle represents the segment's start frame, and the right hand edge its end frame. Segments' annotations are then associated with their graphic representation in some way, and users may request that segments are presented (usually by using a mouse to double click on the required segment).

Approaches such as Harrison *et al.*'s Timelines [89], Mackay *et al.*'s DIVA [130], Kazman *et al.*'s Jabber [108], and Davis' Media Streams [55] represent different kinds of annotations on different *tracks* of the time line. Figure 10 shows an example taken from Harrison *et al.*'s work. There are four kinds of annotation in this figure, each represented on a separate track: Scene change, Narration, Cut, and Music Playing. As Scene changes and Cuts are regarded as single frame events, they are represented in this system by ovals. This is because rectangles representing single frames might be difficult to visualise in the interface. Davis and Kazman *et al.* follow a similar approach, except that the tracks are labelled with icons rather than simple text. Mackay *et al.*'s approach differs in that the tracks are displayed in perspective, similar to the display of micons.

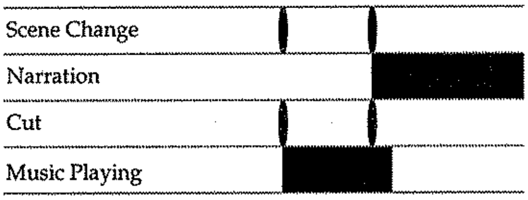


Figure 10: Example of a track based time line

In contrast, Auguierre Smith *et al.*'s Stratagraph [12] imposes no track based structure on the representation of segments. In their system segments are represented by lines; segments' textual annotations are located above these lines. Figure 11 illustrates such a free form time line. This example is taken from the description of the Stratagraph, and shows several overlapping segments and their annotations describing news footage of a "Rescue After the San Francisco Earthquake".

Time lines essentially provide a fine grained overview of the temporal relationships between segments; they represent the way models' segments temporally overlap each other. However, although they allow scaling of the temporal dimension [12], or an

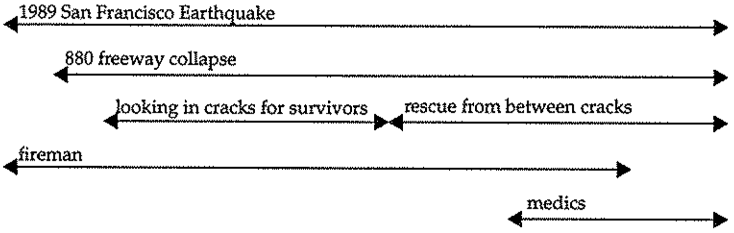

1989 San Francisco Earthquake
880 freeway collapse
looking in cracks for survivors
rescue from between cracks
fireman
medics

Figure 11: Free form time lines

overview of the time line itself [89][108], they usually only provide an overview of a small amount of video. To this end they could be used in the analysis of video at a lower level of granularity than video presented in representative frame based overview systems.

- **Spatial layout of video's temporal dimension: a textual overview**

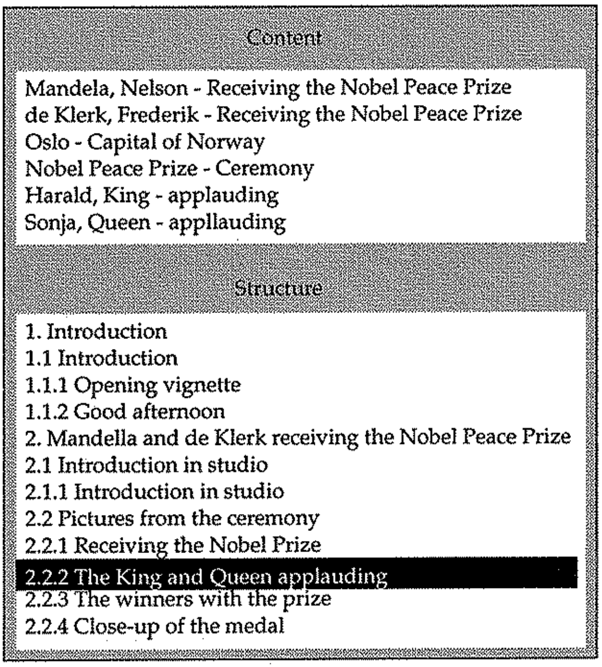
  Geißler [75] provides a textual overview of video. Annotations of atomic segments (shots in this case) are presented in a list to form a textual abstract of the video's content. Hjelsvold *et al.* [92] and Ho-Shing *et al.* [94] also describe approaches which present text only overviews of video documents. In contrast to Geißler, they use annotations of segments *and* the hierarchic relationships between them. These are used to present an interface which is similar to a table of contents in a book (rather than an abstract). Figure 12 shows an extract from Hjelsvold *et al.*'s Video Document Browser. The top box of the interface shows details of the segment which is currently being presented. The bottom box shows the hierarchic structure of the video document. The highlighted text in this box is the current segment, clicking on a line of text causes the corresponding segment to be played.

- **Representation of hierarchic relationships: the movie catalogue**

  Geißler makes use of the hierarchic relationships between annotations in other overview interfaces. Movie Catalogue provides an interface which replicates the Apple Macintosh Finder; windows contain segments at the same level of hierarchy. These segments are represented by icons (containing small versions of the segment's representative frame), or in a textual list (using the segments' textual annotation). Segments containing other segments are represented using folder icons, atomic segments are represented by film strip icons. The hierarchy can then be navigated using the usual Macintosh Finder metaphor by double clicking on folders. Similarly, atomic segments can be viewed by double clicking on them. However, strict adherence to this metaphor means that only atomic segments can be viewed. This means than in the film based hierarchy, only shots can be viewed; whole scenes or films cannot be viewed.

- **Spatial layout of video's temporal dimension: the movie map**

  Another overview tool unique to Geißler's work is the Movie Map [75]. This provides



Content

Mandela, Nelson - Receiving the Nobel Peace Prize
de Klerk, Frederik - Receiving the Nobel Peace Prize
Oslo - Capital of Norway
Nobel Peace Prize - Ceremony
Harald, King - applauding
Sonja, Queen - appllauding

Structure

1. Introduction
1.1 Introduction
1.1.1 Opening vignette
1.1.2 Good afternoon
2. Mandella and de Klerk receiving the Nobel Peace Prize
2.1 Introduction in studio
2.1.1 Introduction in studio
2.2 Pictures from the ceremony
2.2.1 Receiving the Nobel Prize
2.2.2 The King and Queen applauding
2.2.3 The winners with the prize
2.2.4 Close-up of the medal

Figure 12: Extract from the Video Document Browser [92]

an overview of the segments of a video document using the preview metaphor of word processors. Figure 13 shows an example of such an overview. Three levels of film based hierarchy are shown in this diagram: part, scene, shot. Shots are the lowest level of the hierarchy and are atomic segments. They are represented in the diagram by small boxes. The rows of boxes represent groups of shots; scenes. Parts are visualised by the vertical grouping of scenes (indicated by the vertical black rectangles). An interesting interface feature of the Movie Map is its ability to represent user's *markers* in the overview. These are represented by pins which the user can set as necessary. Also of note is the history support; users can tell which shots they have viewed as their boxes are filled (as opposed to empty). However, although the user is informed of the currently playing video segment (by the box surrounding the appropriate shot box), they have no idea of the segment's length, nor any of its annotations. Moreover, the between segment hyperlinks supported by Geißler's system are not represented in this overview.

In summary, models which are be used in such forms of overview need to represent: segments, representative frames, textual annotations, structured annotations, typed annotations, and some form of hierarchical relationships between annotations.

2.6.8c Finding segments within a video document: The query process

There are essentially two forms of querying a video: the model of the video can be queried, or the video itself can be queried in some way. The first form of querying is of relevance to this thesis as it concerns the querying of models. A user poses a query and a collection of matching annotations (and their segments) are located. These can then be

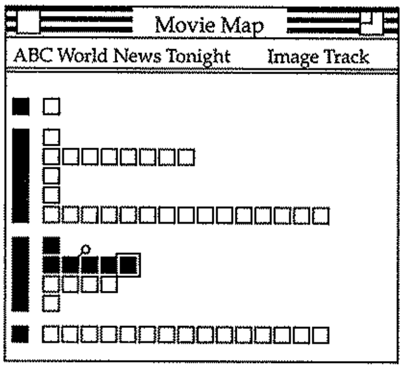


Figure 13: Example of Geißler's Move Map [75]

viewed, browsed, or iteratively queried. The latter form of querying is not considered by this thesis as it involves image analysis and matching, for example locating video frames which match a frame selected by the user. Approaches to this form of querying, such as CORE [205] (Content Based Retrieval Engine), have typically developed from still image matching research (*e.g.* [95][151][184][153]); they match single frames of interest in the video and make no use of segments or annotations in the video's model.

- **Image matching to support queries**

Image matching research is also used by some approaches to querying models. Typically these approaches attempt to match representative frames of a model with a frame selected by the user. This has advantages over simply querying all frames of a video as the representative frames are intended to provide static representations of segments. This makes querying less computationally expensive as one representative frame may represent tens or hundreds of frames in a segment. However, it may be a double edged sword. On one hand a representative frame is intended to capture the important visual information from a segment which prevents less representative frames from being selected from the segment. On the other, there may be images in the segment which are not captured in the representative frame, but which the user would still wish to be located by the query. Three kinds of image matching are considered [213]: template manipulation, object feature specification, and visual example.

- **Template manipulation** [213] involves the user defining a spatial template of the layout of colour patterns. For instance, the spatial template which matches frames of a desert may contain yellow at the bottom of the frame to match the desert sand and blue at the top to match the sky. This relies on the assumption that items of interest will have a similar spatial layout of colour in the frame. This querying approach uses techniques similar to those used in the index sub-process to classify segments, for example, the Zhang *et al.*'s news segment classification approach [212]. Indeed, these approaches have researchers in common, but the news classification approach uses the technique to generate annotations for classification purposes rather than to support querying of annotations.

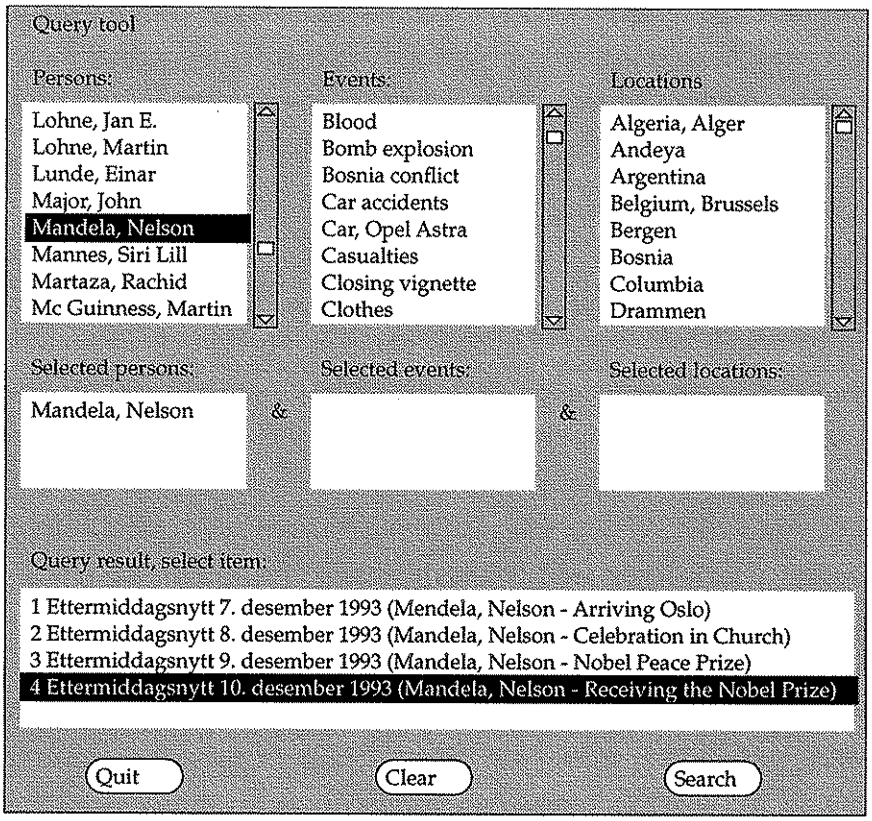
- **Object feature specification** [213][203] allows a user to specify certain features of objects to be matched by a query. For example, locating representative frames with objects whose spatial features (determined by performing edge detection on identified objects) resemble those of a human. This assumes that the object to be located has recognisable features which can be defined by the user.
- **Visual example** [213][11] retrieves segments whose representative frame visually matches (using image matching techniques such as those described by Chua *et al.* [43]) a frame selected by the user. As with other matching techniques, different levels of sensitivity can be selected. Arman *et al.* [11] also provide a visual indication of the match between the selected frame and the key frames located by scaling the image to indicate the match.

- **Queries of textual annotations**

Approaches which do not use image matching techniques rely on textual annotations of segments to support their queries. The simplest form of textual query allows the user to specify a piece of text to be located in an annotation, and to determine which kinds of annotation should be searched [165][123][75][208][101]. For example, locating segments in which a particular actor appears involves searching annotations which contain details of actors for the actor's name.

In order to improve the matching of textual annotations, some approaches use information retrieval techniques to *condition* the search and annotation text. Conditioning involves removing words such as 'the' and 'a', and reducing remaining words to their stem, for example, 'managing' 'manager' and 'manage' all become 'manag' [24]. Other approaches attempt to identify keywords and key phrases in the annotations, and then support querying of these keywords and phrases [179]. Kazman *et al.*'s Jabber [108] uses *lexical chaining* to recover the semantic clusters of words from video recordings of meetings. These clusters are then regarded as *themes* which can be used to support theme based information retrieval rather than word based retrieval. In order to support a wider range of queries of textual annotations than those evident in current video content modelling literature other information retrieval techniques could be applied *e.g.* those discussed by Ginsberg [78] and Frisse *et al.* [65].

A further approach to improving query results is to only allow the user to construct queries containing items in the annotations. For example, Hjelsvold *et al.*'s Integrated Video Archive Tools [92][93], and Duda's Algebraic video [59] use query interfaces in which attributes of annotations are displayed along with their possible values. The user can then formulate their query from attributes and values that actually exist within the annotations. Similarly, Adah *et al.*'s Advanced Video Information Systems [1] allows queries to be formulated from particular objects or events (represented by annotations in a video content model) that exist within the video document. Figure 14 shows a reconstructed example of Hjelsvold *et al.*'s Integrated Video Archive Tools being used to formulate a query. The top three boxes show lists of possible values for different types of annotations. The middle boxes show what has been selected to find, and the lower box lists matching annotations which the user can then view. In this query the user has submitted a query to find annotations containing the person 'Mandela, Nelson'. This reduces the problem of queries not finding appropriate annotations, but large lists of possible values may confuse the user. This approach also provides a good illustration of the use of typed and structured annotations in models.



Lohne, Jan E.
Lohne, Martin
Lunde, Einar
Major, John
Mandela, Nelson
Mannes, Siri Lill
Martaza, Rachid
Mc Guinness, Martin

Blood
Bomb explosion
Bosnia conflict
Car accidents
Car, Opel Astra
Casualties
Closing vignette
Clothes

Algeria, Alger
Andeya
Argentina
Belgium, Brussels
Bergen
Bosnia
Columbia
Drammen

Mandela, Nelson

1 Ettermiddagsnytt 7. desember 1993 (Mendela, Nelson - Arriving Oslo)
2 Ettermiddagsnytt 8. desember 1993 (Mandela, Nelson - Celebration in Church)
3 Ettermiddagsnytt 9. desember 1993 (Mandela, Nelson - Nobel Peace Prize)
4 Ettermiddagsnytt 10. desember 1993 (Mandela, Nelson - Receiving the Nobel Prize)

Figure 14: Hjelsvold *et al.*'s query formulation interface [92][93]

- **Querying segments' temporal properties**

The temporal characteristics of segments can also be included in a query. Hibino *et al.* [91] describe a query language called TVQL (Temporal Visual Query Language) in which the temporal relationships between segments can be included in queries of annotations and their attributes. Using TVQL users can specify queries such as 'show me all annotations where the student is working *while* the teacher is speaking' [91]. This provides powerful querying capabilities which exploit the temporal nature of video and the temporal information contained in models. However it requires the use of a specialised query interface as natural language interpretation of such statements is difficult.

In summary, querying models of video requires that they contain segments, and some relationships and/or annotations which can be queried. These annotations are either queried visually when they contain representative frames, or using some sort of key word or textual query when they contain text. Structured textual annotations are important in this process as they allow the user to specify which particular kinds of information they wish to query rather than having to query all annotations.

2.6.9 Making Video Available for Reuse in New Contexts: The Reuse Process

This thesis bases it definition of *reuse* on Garzotto *et al.*'s definition [74] which is itself derived from software engineering literature. It considers reuse to be the use of data from one *context* in another. A context is a data item's relationship to other data items. For instance, the context of a segment in a video document is its neighbouring segments. Similarly, the context of an annotation is its associated segment, and annotations that are related to it. This is the context of reuse *within* an application (the *data* context). Reuse can also take place *between* applications (the *application* context). In these cases the context of application has changed as well as the data item's relationship to other data items. Reusing data in different applications often leads to different interpretations of the same data in the new application context.

Reuse is categorised in three ways:

- **Whether the new context is in the same, or different application of video.** For example, a segment (such as a television logo) may be reused in the same video document; this is reuse in the same application. Conversely, video may be reused in a different application of video. For example, a segment from a film may be reused in a documentary. Note that in this example the segments are reused in a different application of video, but the video itself may be structured using the same scheme *e.g.* a scheme which structures video using a film metaphor, such as the scheme used in MAD [15]. Sometimes segments are reused in a different application which uses a different modelling scheme. For instance, footage from an interactive documentary such as ConText [53] may be copied into a conventional video document (which would be structured using a conventional film based scheme).
- **What information is being reused.** In video usage this can be: a scheme, annotations (and their associated segments), or the video itself.
- **How the information is reused.** Information can be referenced in its new context, or copied into its new context.

Two forms of reuse take place in this paradigm: reuse of segments in the edit stage to make different versions of the same video document, and reuse of video from a video archive in a new video document. The former is an example of reuse of segments by copying into a new context in the same application. The latter is an example of reuse of segments by copying into a new context in a different application of video.

Conventionally in the reuse process the required segment must be identified in the video document and then included in a new sequence in the edit process by copying and cutting. For instance, including stock footage (short pieces of video whose copyright can be purchased) from a supplier such as *FilmBank* in a documentary involves locating the stock footage in the locate process, retrieving it, making a copy of it, and finally inserting it into the new sequence in the sequence sub-process. This is time consuming, laborious, and difficult [203]. Even if the editing and archival is supported digitally, video of interest is usually copied onto cassette from the archive and then copied into the digital editing suite in the edit process. However, such forms of reuse *are* employed in the conventional paradigm and are often more cost effective in terms of time and labour than capturing new video.

Some systems support reuse of video in this paradigm. For example, Video Mosaic [129] and IVAPS [48] support the reuse of video in the edit process. Different sequences of video can be created, reusing common segments. These different video documents are then distributed for different purposes *i.e.* the application context is different. For example, a product presentation might have different video documents for the marketing department and the board of directors.

However, reuse of video in this film based paradigm is restricted in comparison to the reuse supported in the paradigm described in Section 2.7. This is because the video must be passed from process to process, and is usually transferred as video documents (rather than the segments which are to be reused), typically without any model.

Most systems interested in the reuse of video use the new paradigm described in Section 2.7. Video reuse is inherent in that paradigm as video is not stored and transferred as whole documents. However, even some systems in the new paradigm perform reuse by copying. This typically occurs when video is imported from external sources, even if they are a different instance of the same system. For example, video reused from other sources in Avid's digital newsroom [14] must be copied into the newsroom system and re-segmented and re-annotated. This copying still occurs if the external source is another digital newsroom.

Reuse is important because video is rich in content and widely used, but expensive to capture, store, analyse, and retrieve [59], and so should be reused where possible. The main problem with reuse in this paradigm is that it is supported by copying segments from one video document to another which does not reduce storage space. It also means that the original annotations and relationships of the reused segments are lost *e.g.* information such its original creator, purpose and context, *etc.* Such information may be useful when browsing or searching the completed video documents in the archive process.

2.6.10 Summary of the Conventional Paradigm

This section presented a survey of video usage in the conventional paradigm (developed from film construction). This paradigm typically involves the creation of ideas and plans, followed by capture or generation of video. This video is then edited (possibly with reused video) to create sequences of video regarded as video documents ready for distribution. Distributed video documents are either viewed immediately or indexed prior to archival or viewing. Indexing itself is an iterative process relying on different forms of viewing to help index video documents and their content. Storage of video in an archive results in a volume of video documents from which video documents can be located for further viewing, or reuse in new video documents. This review also highlighted that many uses of video which do not immediately appear to fit this paradigm do so after closer inspection.

Models and schemes used in each process of this paradigm have been highlighted and discussed. Such models support the two main uses of video: analysis and presentation. Models described have shown different levels of granularity of annotation, different kinds of segmentation (segments or clips), different kinds of annotations (from free text to structured data types), and different kinds of relationships (mainly hierarchic, though other relationships such as sequential and association relationships were also reviewed).

A major problem with this paradigm is that video documents are typically copied between processes (regardless of the storage medium). This is usually achieved *via* some intermediate storage medium such as a video cassette. Using an intermediate storage medium results in models not being shared between processes. It also means that storage requirements are increased and image quality is decreased as every process and path between processes require their own copy of the video document. Multiple copies of video documents leads to the inconsistency of models, and the need to recreate information in other models *e.g.* the re-segmenting of video documents in the index process.

The next section discusses a paradigm for video usage which significantly differs from this paradigm. The same processes can be applied, but they are applied in different ways, often with different results. In terms of video content modelling, this new paradigm integrates the models more closely with their related video and no longer regards the primary unit of information as the video document.

2.7 The New Paradigm

The conventional paradigm of using video is a direct development of the process of using film, and film making. As such the end product is a *linear* video document where the single sequence of video segments is reflected by their sequence in the storage medium. As the storage medium reflects the sequence of segments no model is necessary for presentation. Innovative uses of video, which often provide interactive, non-linear navigation through video [121][173][50][52][53], have moved the emphasis from producing linear video documents to producing video documents whose presentation is determined by its model (rather than the storage medium). This means that an explicit model of the video document is essential to using video in the new paradigm; the model determines the possible sequences of segments and can be used to interactively locate and manipulate video.

Figure 15 illustrates this unconventional paradigm of video usage. At the centre of the figure is the store process which stores video and its associated models in some form of database. Davenport *et al.* [53] describe this as a new type of database; a 'Media Bank'. Instead of regarding completed video documents as physically separate entities (such as feature films), all video is stored together in one repository; the models define how video is played back. The most extreme interpretation of this would propose that video frames (regarded as the smallest unit of video which is usefully indexed) do not need to be stored in any sequence; segments are ordered collections of frames. However, in current research [121][173][50][52][53], video frames are still stored in a linear sequence meaning that segments can be defined by their start and end frames. But segments are not necessarily stored in their presentation order. Moreover, the model usually defines more than one possible segment presentation order.

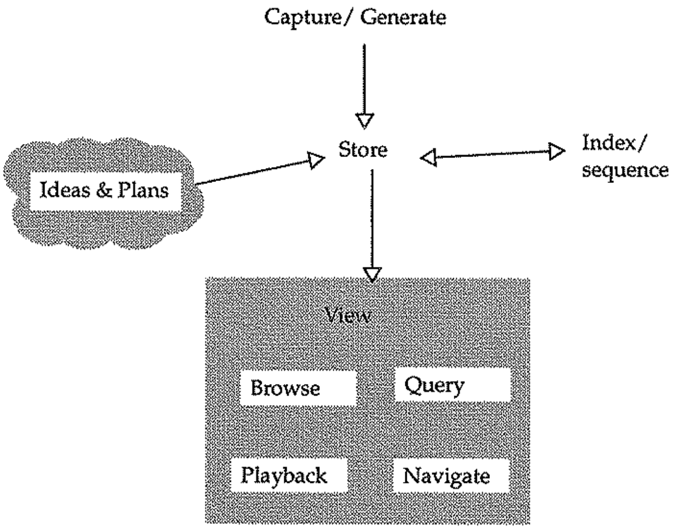


Figure 15: The new paradigm to video usage

The central repository in this paradigm means that models of video created by one process are available to others *via* some sort of video database management system. This contrasts with the conventional paradigm in which models are often non existent or lost between processes. It also means that reusing video is much easier as it does not need to be copied into its new context; a model defines the new sequence to be presented. In fact, there is no reuse process in this paradigm as it is implicit in the way the video and its models are used. In terms of the definition of reuse, this means that segments (and their associated annotations) can be reused by referencing them in the new context (in the same or different application). The problem with reuse by referencing is that it may be difficult to adjust the segments in their new context *e.g* cutting them to fit the new context.

In comparison to the conventional paradigm described in the previous section, similar operations are carried out on the video and its models. However, they are typically carried out in different ways, and aim to produce video documents which contain multiple sequences of segments determined by models. Furthermore, these models may be influenced by viewers during playback of the video document.

Commercial products such as Avid's Digital News Gathering (DNG) system [14] highlight the difference between the conventional paradigm and the new paradigm. Their system aims to support the whole process of news production: recording, storing, and cutting footage, creating sequences for broadcast, and archiving completed sequences. Although they still follow the workflow of the conventional paradigm (capture, edit, distribute) the video documents produced (news items) rely on a model for their sequence *i.e.* the sequence of segments in news items is not necessarily determined by the storage medium. Moreover, the news programmes transmitted are composed of several news items which may be selected or changed in real time as the programme is transmitted (typically by the director). Unlike news broadcast systems using the conventional paradigm, changing the sequence of news items in the new paradigm involves changing the model rather than manually loading different video cassettes for presentation.

In keeping with this new paradigm reuse is implicit in Avid's system. Segments can easily be reused in other news items, and news items themselves can be reused in different news programmes. Interestingly Avid's system also supports the creation and storage of scripts in the Ideas & plans process. These scripts can be related to the appropriate segments of video to provide some form of annotation. Also in keeping with the new paradigm, their system uses random access storage where possible; only archived material is stored on sequential access media.

However, it should be noted that Avid's approach regards each news producer as a separate entity. Video documents are typically imported from other producers using tape or fibre optic connection [14] and so have no model associated with them. They must be re-segmented and annotated in their new context. As such Avid's approach supports reuse by reference locally, but reuse by copying from external sources.

It should also be noted that although some new developments of video database management systems have the underlying technical support for the new paradigm they still regard video in terms of the conventional paradigm. That is, although the video frames could be accessed dynamically, and in any order, the sequence of video frames presented is defined by the sequence of video frames in the medium. For example, Yeo *et al.*'s work [208] is concerned with the retrieval of video from a video database, and the reuse of located video in new sequences. However, they consider a video document as a single linear sequence of video segments which is implicit in the way it is stored. This contrasts Avid's system in which a model determines the possible sequences of segments and allows the dynamic selection of segments as they are presented.

The rest of this section concentrates on the main difference between this and the conventional paradigm; the way a viewer interacts with video. Table 3 provides a brief overview of how the other processes fit into this new paradigm.

2.7.1 Viewing a Video Document: The View Process

Approaches using this new paradigm are no longer restricted to pre-defined sequences of segments. This means that they can provide highly interactive video playback in which viewers can influence which segments are presented. Most work which uses this new paradigm concentrates on new and more interactive ways for viewers to influence video presentation.

2.7.1a Viewing by navigating using a spatial metaphor

Lippman's video map [121] provides viewers with a virtual car journey through a town. In terms of this paradigm, video recordings of car journeys (from a driver's perspective) are recorded and stored as segments in the central repository. The model contains annotations and spatial relationships. The annotations describe the direction of travel in a segment and whether it is a junction or a straight piece of road. Viewers are presented with a visual representation of a car journey through the town recorded. The viewer can change the speed of the journey, and can select which turn to take at junctions, *i.e.* at certain points (junctions in this case) the viewer can decide which video segment the system should present next.

Similarly, Sawhney *et al.* [173][174] describe a system which allows a viewer to navigate around a café. Again, recordings of the various paths around the café are stored in the central repository as segments, the model contains annotations describing their location. Spatial relationships exist between segments in order to support user navigation through the space. The viewer is presented with an interface which allows them to navigate around the café by selecting which table to approach. Additionally, Sawhney *et al.*'s work includes recordings of conversations that the viewer can influence. Sentences from the conversation are stored as segments along with textual transcripts as annotations in the model. Viewers can use sequence relationships to dynamically join segments together into new conversations [173][174].

Some early arcade games use similar navigation metaphors to control the output of video disc players. For instance, *Dragon's Lair* by *Digital Leisure* is an arcade game

Process	Action
Ideas & plans Capture/ generate	Ideas and plans may be generated in the ideas process and then stored in the central repository. Video is captured or generated by the capture/ generate process. This is also stored in the central repository where it can be accessed and manipulated by other processes. Initially this *raw* video will simply be described as existing in the model of the video repository. A segment will define the raw video's temporal extent, and may be annotated with information about when it was captured, by whom, for what purpose *etc.* (information as described in the capture process of the conventional paradigm). These segments themselves could easily be related to their associated parts of the plan.
Index/ sequence	The index/ sequence process defines segments of video and associates annotations with them. This often results in models containing many different interpretations of the same video or segments. This is beneficial as video is very rich in content resulting in different interpretations of the same video being common [55]. As with the conventional paradigm, segments may be defined manually or automatically. However, automatic segmentation may not be as useful in this paradigm as many of the segmenting algorithms in the index process of the conventional paradigm, such as cut detection algorithms [44][85][190][213], attempt to *re*-segment video into its original segments. In this paradigm there is no need to re-segment the video as models are not lost (as they are between the edit, distribute, and index processes of the conventional approach). This process also performs the function of relating annotations to each other. Typically this involves creating hierarchic relationships between annotations, and/or creating sequential relationships between annotations to define a sequence of segments. Again, there may be many different sequences of the same segments as this is defined by the model, not the sequence of segments in the video itself.
Locate/ view	Unlike the conventional paradigm of using video, there is no longer the forced distinction between the locate and view processes. That is, it is not necessary to first locate a video document (such as a feature film) in the locate process, and then locate items of interest within that video document in the view process. Instead, each segment is individually accessible, and its context is preserved *via* the model *e.g.* the fact that a scene belongs to a particular video document is defined by the model, not the fact that it resides in a particular video recording. The lack of distinction between the locate and view processes makes iterative searching and browsing of the video archive easier. This is because users search using one common model, rather than one model in the locate process and another in the view process. This makes the application of information retrieval techniques such as Ginsberg's WorldViews [78] more feasible within a video archive. These techniques typically support users in iterative queries such as those discussed by O' Day *et al.* [145] who found that 'a single search [query] evolved into a series of interconnected searches, usually beginning with a high-level overview'. In the conventional paradigm, such information retrieval techniques are used separately in the two processes. This makes the interconnection of queries difficult as the user must discretely move between one kind of search and another.

Table 3: Overview of processes in the new paradigm

developed in the 1980s in which players navigate through a fantasy landscape to reach their goal. The images of the landscape are presented from video disc; at certain points the user can influence the choice of the next segment by deciding which direction to travel. The rationale for using images stored on video discs is that at the time of development it was not possible to render high resolution graphics in real time and at reasonable cost; they were rendered elsewhere and stored on the video disc.

2.7.1b Viewing by influencing the story told

Work by Davenport *et al.* [50][52][53] (summarised in [17]) concentrates more on viewers *influencing* the segments selected, rather than directly selecting them. The thread of work between Elastic Movies (1989), Digital Micromovies (1993), and ConText (1995) shows how their approach has developed. Essentially the system has a story to tell, and has rules based on story telling, cinematography, and viewer preference that tell it how to present it. In Elastic Movies, the initial ideas behind these approaches were investigated manually: basic rules of cinematography were used to generate a set of sequences of the same story. In Micromovies various combinations of *filters* were used to provide interactive control of a story. For example, a combination of three filters might work as follows:

1. A *continuity template filter* selects segments that are appropriate for the current point in the story. For example, this would ensure that question segments were followed by appropriate answer segments.
2. A *stylistic filter* would then be applied to the remaining segments. This uses conventions of film making to enhance the viewer's experience *e.g.* 'if the character was seen in the last shot in extreme close up, then do not show her in extreme close up in the next shot' [52].
3. Then an *interaction filter* selects the most appropriate segments from the remaining set. This filter is controlled by the viewer *e.g.* *An Endless Conversation* [52] allows the viewer to change the pacing of the presentation from slow to fast during playback.

Recent work by Davenport *et al.* involves *evolving documentaries* [53]. Again, the viewer influences the presentation of video (documentaries in this case). Here though the viewer is presented with several segments of video at the same time. Segments are annotated with information about the topic(s) they contain, and a prominence value which is used by the system to determine which segments are of interest to the viewer. The viewer indicates the most interesting segment by moving their mouse over it. This causes the system to give that segment, and its annotated topic high prominence. The system then uses cinematic and story based rules to present video segments to the viewer in increasing detail about that topic, and, to a lesser extent, related topics. At any point the viewer can express their interest in another displayed topic which causes the system to start concentrating on other segments. Another difference is the fact that the central repository of the ConText system can be added to at any point as new footage is collected (hence the term *evolving* documentary). This means that a viewer can revisit the same topic at a later date and be presented with new footage.

Although these systems use rules to create sequences, they are different to the automatic sequencing approaches discussed in the **sequence** sub-process of the previous section. These rules are used to determine the next segment to be presented, rather than to construct whole sequences. Moreover, the viewer can interactively influence the decisions taken by these rules.

2.7.2 Summary of the New Paradigm

The models in this new paradigm contain segments, annotations, and relationships. With respect to the conventional paradigm, models in this process make much greater use of relationships between annotations. Sequential relationships are used instead of assuming that the segments are sequenced by the storage medium. Spatial relationships are used to describe the relationships of the spaces recorded in the segments [121][173]. Topic relevance relationships relate segments in terms of the relevance of their annotated topic descriptions [53]. This paradigm subsumes the conventional paradigm; anything that could be done in the previous paradigm could be done in this paradigm, but not *visa versa*. Moreover, performing tasks involving video using this paradigm can be beneficial as models are not lost between processes, and so do not have to be recreated, reuse is inherent in the paradigm, multiple interpretations of the same video are easily handled, and there are more interactive possibilities.

However, the approaches discussed in this section essentially regard segments as clips (see Section 2.1 for a discussion of the difference between segments and clips). This means that there cannot be overlapping segments which are essential to many uses of video, such as the analysis of behaviour discussed in Section 2.4, and highlighted by the timeline interfaces discussed in Section 2.6.8. HyperCafé [173] allows relationships to point to frames within segments, but segments are still regarded as non-overlapping. To fully satisfy the requirements of video content models laid out in Section 2.1, these approaches need to be able to describe segments rather than just clips. This reflects the immaturity of this new paradigm when compared to the conventional paradigm.

A further problem with approaches in this new paradigm is that they cannot share models between each other, even if they are instances of the same system. Avid's digital newsroom [14] highlights this problem. Video from external sources such as other news rooms has to be copied into the system and re-segmented and re-annotated. Even if video is to be shared between two of the same systems an intermediate medium is used. This intermediate medium is typically sequentially accessed and lacks any model *e.g.* video cassette.

2.8 Characterisation of Properties of Video Content Models

This review has surveyed many different kinds of models and schemes. In terms of the characteristics of models discussed in Section 2.1, the following characteristics are evident:

2.8.1 Kinds of Segments

The list of aspects of video content which need to be represented in video content models (described in Section 2.1) are evident in the kinds of segments in the models surveyed:

- Segments representing video *documents*. Systems such as Rowe *et al.*'s Video Database Browser [165] annotate segments representing video documents with information about their title, bibliography *etc.*
- *Clips* defined within video documents. Systems such as Davenport *et al.*'s ConText [53] define clips of video and annotate them with descriptions of their content and relevance to current viewer interest.
- *Segments* defined within video documents. Approaches such as Duda *et al.*'s algebraic video [59] support the definition of segments of video and their associated annotations.

2.8.2 Annotations Evident in Video Content Models

To recap, an annotation is a piece of data associated with a segment whose purpose is to provide some description of the segment's content. The models reviewed show many different types of data used as annotations:

- **Free text.** Some systems support free text annotations. That is, the annotation can contain any text the user wants *e.g.* the annotation supported by Auguierre Smith *et al.*'s Stratagraph [12]. Most systems support more structured annotations as free text annotations can lead to ambiguity and inconsistency [44].
- **Keywords.** These provide a more regulated form of annotation, and so reduce possible inconsistencies and ambiguities. Work by Shahraray *et al.* [179] exemplifies the use of keyword annotations, and their use in supporting querying of video.
- **Representative frames.** Many of the systems reviewed in this chapter have some notion of representative frames of segments. These are usually realised by storing a frame number in the annotation, and retrieving the frame as necessary [44].
- **Typed annotations.** Most approaches support typed annotations, *i.e.* annotations of a particular type describe content of a particular type. Most systems supporting typed annotations base their type system on the film metaphor; annotations can typically be of type: sequence, scene, or shot [182]. Alternative typing systems include those developed by Zhang *et al.* [212] to describe news broadcasts. In their type system annotations can be of type: header, anchor person, news reel, or weather.
- **Structured annotations.** Some of the systems in this chapter allow annotations to be structured in some way. For example, Chua *et al.*'s VRSS [44] represents annotations using *frames* [44]. A frame is a data structure which contains *slots*. Each slot has a name and can take a value of a specified type (including other frames).

2.8.3 Relationships Evident in Video Content Models

The most prevalent relationships used in the video content models reviewed in this chapter are hierarchic relationships. These are typically used to represent the conventional hierarchic structure of films in which films are made up of scenes which are in turn made up of shots [182]. Other kinds of relationships, such as sequence and association, are also evident in these models. A taxonomy of relationships which encompasses the relationships discussed in this chapter is presented here. This is derived from De Rose's [56] description of a taxonomy of links (relationships) for hypertext which can be considered a forerunner of hypermedia and so is pertinent to this related field of video content modelling. The taxonomy extends De Rose's work by taking into account the temporal nature of video, and the needs of relationships between complex media such as video (discussed by Garzotto *et al.* [70][71]). This temporal nature means that greater emphasis needs to be placed on the relationships within groups of annotations; De Rose's work is based on links between static media and so only considers linear sequences within groups. However, De Rose's work is pertinent as it provides a taxonomic framework within which the relationships discussed in this chapter can be distinguished and discussed.

Figure 16 illustrates the taxonomy of relationships. This taxonomy assumes that all relationships are stored in some way. Even the intentional (calculated) links of De Rose, such as the links created by a search, are regarded as being stored in a model as this thesis is interested in models, and the relationships stored within them.

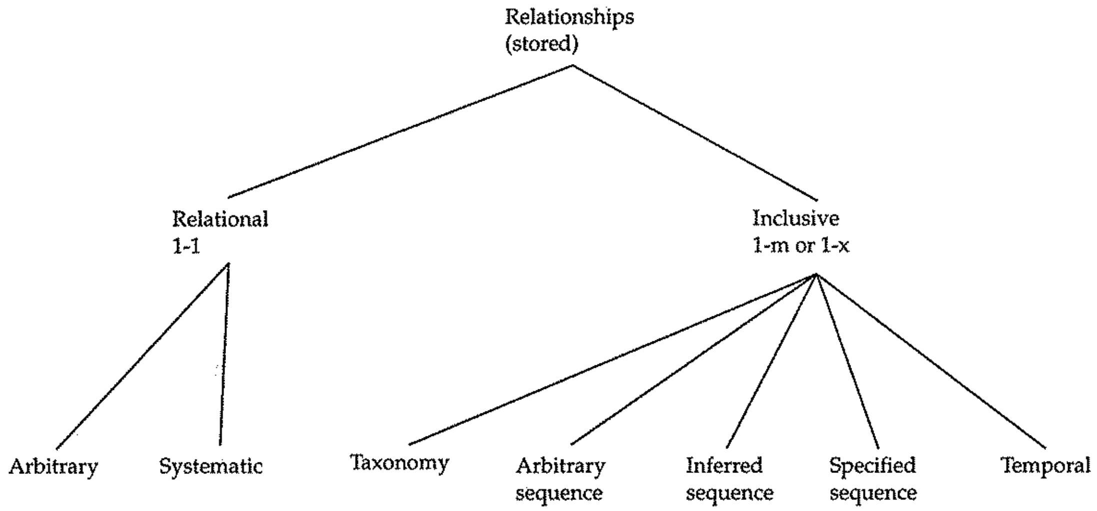


Figure 16: Taxonomy of relationships derived from De Rose's taxonomy [56]

De Roses's taxonomy categorises relationships into *relational* and *inclusive* relationships. This distinguishes between relationships which relate one annotation to another, and ones which relate one annotation to many. There is a trade-off at this point between the utility of such a distinction and its generality; this distinction is general, but its utility lies in distinguishing the underlying constraints of the relationships before the taxonomy provides a semantic categorisation. The taxonomy is as follows:

2.8.3a "Relational" relationships (1-1)

These relationships relate one annotation to another. This is either done purely *arbitrarily* (for example the association relationships supported by Video Mosaic [129]) or in some *systematic* way whereby the kind of annotations to relate are known, but the actual annotations must be specified by the user in some way. Movie maps [121] use such systematic relationships when constructing the map. The scheme states that each straight road must meet a junction, but the author must decide which road meets which junction.

2.8.3b "Inclusive" relationships (1-m or 1-x)

These relationships relate one annotation to one or more others (1-m *e.g.* in the film based metaphor, one scene is related by structural composition to many shots), or to a specific number of other annotations (1-x where x > 1). They are used to describe collections of annotations. Different collections (which may contain heterogeneous or homogeneous types of annotations) relate their member annotations together in five different ways (compared to the two ways described by De Rose for static media):

- **Taxonomy**. There are no relationships between the members of a collection. Such relationships are evident in systems which provide indexes of video documents. For example, Rowe *et al.*'s genre hierarchy [165] can be represented using taxonomic relationships; there is no ordering between the members of classifications.
- **Arbitrary sequence**. The user defines relationships between the members which represent a linear sequence. Systems which support user sequencing of video, such as MAD [15], exemplify the use of arbitrary sequence relationships; the user can construct sequences of video segments by sequencing their annotations.
- **Inferred sequence**. Relationships exist between the annotations to indicate a sequence inferred from the content of the annotations. Many systems, such as Rowe *et al.*'s Video Database Browser [165], use inferred sequence relationships. Typically the sequence of shots, or scenes in a movie is inferred from their start and end frames *i.e.* part of the content of the annotations (assuming that the start and end frames are stored in the annotations). De Rose considers relationships of this kind to be 'intensional' as they can be inferred from the annotations.
- **Pre-defined sequence**. The members are sequenced in a pre-defined way using relationships. Approaches such as Swanberg *et al.*'s knowledge guided parsing [186] use pre-defined sequence relationships to construct finite state models of news broadcasts.
- **Temporal**. The temporal relationships between segments are reflected in the relationships between the associated annotations. Approaches such a TVQL [91] use the temporal relationships between annotations (and their segments) to support querying about their relative temporal positions. These relationships are different to inferred sequence relationships as there are several different kinds of temporal relationships (start, meets, overlaps, *etc.*) as opposed to the linear sequence described

by inferred sequence relationships.

2.8.4 Uses of Video Content Models

The survey highlights the three main uses models are put to: analysis, *deterministic* presentation, and *non-deterministic* presentation. The systems reviewed highlight the heterogeneity of uses of models. Models do not typically have one use (*e.g.* solely used for analysis), but they do usually have a main use (*e.g.* mainly used for analysis, but also able to be used for presentation). Models are used for:

- **Analysis.** Systems which are primarily used for analysis concentrate on constructing models of the content of video. The models themselves are analysed to infer something about the content of the video. For example, EVA [128] and DIVA [130] support the construction of models of the content of video recordings (primarily observational recordings). These models than can then used to test or construct hypotheses about the behaviour recorded in the video. Models primarily used for analysis tend to use inferred, pre-defined, and temporal relationships between members of collections. This is because analysts are typically interested in examining the relationships between items of interest in the video (reflected in the model).
- **Presentation (deterministic).** Deterministic presentation systems concentrate on the construction of linear video documents. Conventional film production constructs video documents in this way. Systems such as Video Mosaic [129] support the construction of such video documents. This type of presentation is referred to as deterministic because the video document produced has a pre-determined length and sequence of segments. The viewer may adjust the rate of playback, but cannot change the sequence of segments in the video document. Models primarily used for deterministic presentation usually use arbitrary ordering of members of collections. This allows the author of the video document to define the order in which the segments are presented to the final viewer.
- **Presentation (non-deterministic).** Non-deterministic presentation, on the other hand, involves the interactive sequencing of segments. There is typically not one pre-determined sequence of segments (though there may be a default sequence); in some way the viewer can influence which segments is to be presented. Approaches such as Davenport's ConText [53] and Lippman's Movie-Maps [121] are examples of non-deterministic presentation as the user can influence the next segment to be presented. Models used for non-deterministic presentation usually use taxonomic collections where there are no sequence relationships between members of collections. This is because users influence the choice of the next segment to be presented.

In summary, models can be differentiated in terms of the following characteristics:

- **Kind of segments**: clips or segments.
- **Kinds of annotations**: free text, representative frame, structured data, typed annotations *etc.*

- **Kinds of relationships**: relational or inclusive.
- **Collection members**: heterogeneous or homogeneous.
- **Primary use**: analysis, deterministic presentation, or non-deterministic presentation.

The relationships used in models can be differentiated in terms of the following characteristics:

- **Relational**: 1-1, arbitrary or systematic.
- **Inclusive**: 1-m or 1-x. Relationships between members: none, arbitrary, inferred, pre-defined, or temporal.

Table 4 presents an overview of some current research systems with respect to these characteristics. This illustrates the different uses and characteristics of models in current use.

2.9 Characterisation of Kinds of Reuse of Video Content Models

The two paradigms discussed in this chapter highlight different kinds of reuse of video and video content models. The conventional paradigm supports reuse by copying segments into new contexts for the same or different applications. The new paradigm supports reuse by referencing which means that there are no duplicate copies of video, and also tends to mean that annotations of segments are retained in their new contexts. Both paradigms implicitly support the reuse of schemes. Schemes themselves are embedded in the systems, and so with each new video (such as the creation of a video document) they are effectively reusing their inbuilt scheme to structure the model and therefore the video document.

The main problem with reuse in these paradigms is the reuse of segments and their annotations from one application of video to another which uses a *different* scheme (a different way of structuring models of video). This kind of reuse generally occurs when video is reused in a different system. The conventional paradigm supports reuse of video by copying segments which typically lose their annotations in the process. In this case the appropriate video is copied without any annotation (and hence without any relationships) into the new context. Although the new paradigm supports reuse by referencing, the annotations of one scheme are typically not understandable by another. This means that even the new paradigm restricts reuse between applications which use different schemes *i.e.* reuse of segments with their annotations and relationships is restricted to the system that created the model.

2.10 Problems Evident in Current Literature

The main problem found in current video content modelling literature that this thesis

Author and system/ aim	General description of approach	Seg'tion[a] and use[b]	Kinds of annotations	Kinds of relationships
Auguierre-Smith *et al.* [12]: Stratosphere.	Multi layered annotation of video, and some reuse of video in new presentations.	S A	Free text.	Sequence relationships for new presentations.
Baecker *et al.* [15]: Movie Authoring and Design (MAD).	Integration of scripts and storyboards with video in the creation of video documents.	C PD	Free text, graphics, sound.	Hierarchic and sequential.
Brown *et al.* [24]: News story retrieval.	Retrieval of news video using the teletext subtitles and information retrieval algorithms.	S A	Free text.	None.
Cherfaoui *et al.* [38]: Indexing video.	Segmentation of video into shots, sequences, and themes. Indexing video by detecting certain objects automatically.	C A	Typed to reflect hierarchy, and structured to describe camera and object (face) movement.	Hierarchic; 3 levels: theme, sequence, shot.
Chua *et al.* [44]: Video Retrieval and Sequencing System (VRSS).	Designed to provide automated support for the whole conventional paradigm (for a film metaphor): segmenting, indexing, retrieving, and sequencing.	S A&PD	Frames (typed and structured annotations).	Hierarchic; 3 levels: sequence, scene, shot.
Davenport *et al.* [53]: ConText.	Evolving interactive documentaries; the user can influence which topics are presented next, and in what detail.	C PN	Typed: character, time, location. Structured: descriptions and relevance weighting for presentation.	Topic relevance relationships.
Duda *et al.* [55]: Algebraic video.	Construction of video documents using video algebra.	S A&PD	Typed and structured by the user.	Hierarchic and temporal composition relationships.
Fischer *et al.* [61]: Automatic recognition of film genres.	Attempts to determine a video document's genre based its syntactic and stylistic properties.	C&D A	Typed and structured to represent the different properties and their values.	None.
Hjelsvold *et al.* [92][93]: Integrated Video Archive Tools (IVAPS).	Provides a digital archive of video material with query and browsing interfaces to support retrieval.	C A	Free text typed annotations: persons, locations, events.	Four level hierarchy: documents, sequences, scenes, shots.
Lippman [121]: Movie maps.	Provides virtual journeys through a space recorded on video.	C PN	Typed: straight road or junction. Structured: geographic location and heading.	Spatial relationships.
Mackay [128]: Experimental Video Annotator (EVA).	Supports the capture, storage, and analysis of video primarily for experimental analysis.	S A	Free text or keywords (codes).	None.
Rowe *et al.* [165]: Video Database Browser (VDB).	Concerned with the location of video by Video On Demand (VOD) users; supports querying and browsing of the video database.	S A	Typed: Bibliographic, structural, object, keyword. Structured: each annotation type has several attributes.	Four level hierarchy: document, segment, scene, shot.
Sack *et al.* [168]: Assembling video sequences from story plans.	Artificial intelligence techniques are used to plan sequences of video which meet particular aims.	C PD	Structured with free text descriptions.	Sequence relationships define the planned sequence.
Sawhney *et al.* [174]: HyperCafé.	Provide virtual conversations and journeys through a café space.	C PN	Free text descriptions.	Spatial, hierarchic (two levels), and sequence (narrative) relationships.

Table 4: Overview of some current research

a. Segmentation: Documents / Clips / Segments
b. Main use: Analysis / Presentation (Deterministic / Non-deterministic)

concentrates on is:

- How can one approach usefully describe and support the use of different kinds of models and schemes?

This thesis addresses this problem and in doing so directly addresses the following problems identified in this chapter:

- The movement of video from one process to another causes problems in itself. Often annotations, segments, and relationships are lost when the video is passed from one process to another. This is especially true of the conventional paradigm where video documents may need to be copied between processes causing the further problem of inconsistent descriptions of the different copies.
- This problem is compounded by the fact that even in the new paradigm reuse by reference is only supported within the application that created it. This means that reuse between applications must be achieved by copying *e.g.* copying a video clip from one application to another, or copying its annotation. This again causes multiple copies of the same information to be present and causes inconsistency of description between applications. Furthermore, it is difficult to access models of one application in another which reduces the richness of video description. This problem is the main problem that this thesis concentrates on; Chapter 3 discusses this problem in more detail.

This chapter also raised other problems in current literature. Addressing the main problem will provide a sound research basis from which the following questions can be addressed:

- What constitutes a suitable description for a segment; is free text sufficient, or are more structured descriptions necessary? Can keywords provide adequate descriptions? Clearly this depends on the purpose of the description as well as the availability of suitably structured descriptions.
- Similarly, in terms of representative frames, what constitutes a *representative* frame. The simplest approaches choose an arbitrary frame to be representative, but others attempt to apply some heuristics to selecting a meaningful frame. The problems has several aspects: attempting to represent a moving image with one still image, attempting to represent non-visual information (sound and especially dialogue) using a still image, and how the representation is affected by its purpose. The final point is the most overlooked in current research. This is problematic as one segment may require several different representative frames depending on the purpose of the representation.
- How can video segmentation and annotation be automated? Currently only simple segmentation and annotation takes place, often without any regard for the content. The largest effort concentrates on the automatic detection of shot boundaries, but uses of video which are not cinematically based require different forms of

segmentation. This leads to the problem of automatically supporting domain independent segmentation. Moreover, only limited automated annotation currently takes place, either by detecting objects, recognising speech, spatial patterns and movement, or using additional information such as broadcast subtitles. How can more semantic descriptions be inferred from video? Some approaches have moved towards more semantic descriptions by inferring semantics from several different simple descriptions.

- How can relationships between annotations be automatically deduced? Some systems attempt to fit segments into a film based hierarchy of films, scenes, and shots. However, as with other automatic analysis techniques, this is domain specific. How can descriptions of relationships and their constraints be usefully constructed?
- In terms of presentation, how can the temporal dimension of video be usefully presented to, and understood by, the user. Again, this depends on the purpose of presenting the temporal dimension to the user. Is it to inform the user of their current position within a segment? To allow comparison between segments? To support composition of segments for presentation? Each of these purposes could require a different method of presentation.
- In terms of reuse of video segments it is unclear how to reuse part of a segment without simply copying the segment and editing it. Some method of reuse by reference which states that some subset of a segment should be reused needs to be defined. Copying and editing leads to the problems highlighted in the previous two points.

2.11 Summary

This chapter has surveyed literature which uses video content models and schemes. The review explored the models and schemes used by examining the two main paradigms of video usage: the conventional and the new, non linear paradigms. Such a review itself constitutes a contribution to current research; other reviews of literature concerned with video have concentrated on particular domains or processes of use. For example, Cherfaoui *et al.*'s review [37] concentrates on automatic analysis of video. Similarly, Yeo *et al.*'s review [208] has a film based perspective of the video handling processes. In contrast, this review surveyed work from different disciplines, and different processes within the two paradigms of video usage. Moreover, it concentrated on the models (*i.e.* the representations of the content) and their schemes (the structural and semantic regularities of models) rather than particular processes or domains.

This chapter also defined key terms and phrases of the thesis, and has characterised important properties of models and schemes. These characteristics are drawn upon in the next chapter in the definition of requirements of a framework to support video content modelling. They are also returned to in Chapter 6 to support the selection of example models and schemes which are representative in terms of current literature.

The next chapter provides a more detailed account of the main problem that this thesis concentrates on in video content modelling: the problem of describing schemes and models, and suggests ways in which it could be addressed.

Chapter 3 The Problem, Solutions, the Thesis

This chapter discusses the main problem highlighted in Chapter 2; the lack of an approach which can describe different video content models and schemes. Section 3.1 details the problem and the reasons it limits the reuse of parts of video content models, particularly between schemes and applications. Section 3.2 suggests what is needed to address this problem and Section 3.3 discusses how the problem could be solved in general, from a hypermedia perspective, and finally from a video content modelling perspective. Section 3.4 defines the thesis and its rationale. Section 3.5 provides a short summary of this chapter and how it relates to other chapters.

3.1 The Main Problem Raised: The Lack of a Single Approach to Video Content Modelling

The main problem that this thesis concentrates on is: the lack of an approach which can describe different video content models and schemes. Without such an approach video content models and schemes are idiosyncratically created, stored, and used. Moreover, reuse is poorly supported. In particular, there is poor support for the reuse of segments and their associated annotations, especially between different modelling schemes and applications. Reuse is important because: it can reduce storage space, it avoids recapture, it enhances accessibility, and it provides different points of view about the same information [74]. The benefits of reuse are shown in the conventional paradigm discussed in Chapter 2 where even though it is time consuming, laborious, and difficult to reuse video, it is still undertaken and considered an important source of video. Reuse is the motivating factor for this thesis; an approach is defined in this thesis which can describe different models and schemes, and which can be used to support various forms of reuse (some of which were previously poorly supported).

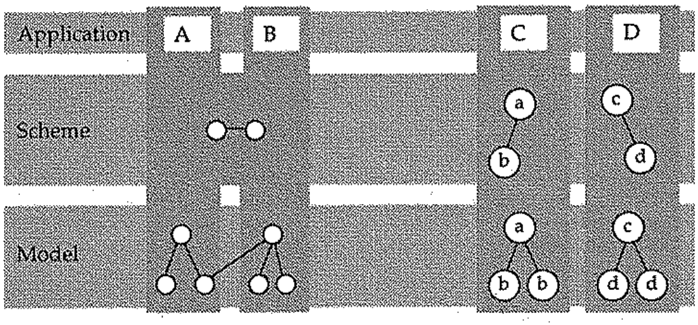
Table 5 illustrates a taxonomy of the reuse of segments and their associated annotations discussed in Chapter 2. Such reuse is of primary importance to the thesis because of its current lack of support. Reuse of schemes (which is also poorly supported) and video itself are considered in Chapter 6. In the taxonomy, reuse of segments and their annotations within the same scheme is supported by most of the approaches reviewed. This is usually supported by copying the segment and its annotation into the new context, or less frequently by referencing them in the new context. Reuse by reference is preferable as it retains access to the original context, and does not require large amounts of extra storage space.

Table 5 highlights the poor support for reuse of segments and their annotations in a model instantiated from a different scheme. This problem is compounded when the new model is used for a different application. The two different reasons for this are:

		Modelling Scheme	
		Same Scheme	**Different Scheme**
Application	**Same Application**	Supported by most approaches - by copying, or by reference.	Supported by some approaches - usually by copying.
	Different Application	Supported by most approaches - by copying, or by reference.	Typically only supported by copying segments (usually using an intermediate medium) without their annotations.

Table 5: Reuse of segments and their associated annotations currently supported

1. Models are instantiated from fixed schemes and so parts of them cannot be reused in models instantiated from other schemes. For example, Rowe *et al.*'s video database [165] uses a fixed scheme in which a film is composed of scenes which are composed of shots. Such a fixed scheme means that segments and their annotations from models based on it could not be reused in a model based on another scheme such as the scheme used in Sawhney *et al.*'s HyperCafé [173]. Figure 17b provides a simple illustration of this situation; the circles represent annotations, the lines represent relationships between them. Letters are used to symbolise different types of annotations. In the figure application C uses a fixed scheme in which annotations of type a can be related to annotations of type b. This is different to application D's scheme in which annotations of type c can be related to annotations of type d. The two applications cannot understand each others' schemes which means that at the model level segments and their annotations cannot be reused between applications.
2. Alternatively, a common scheme is used which means that segments and their annotations can be reused in different models and applications. However, such a scheme is so general that it does not describe the structural and semantic regularities of classes of models. This leads to inconsistent use of segments, annotations, and relationships [71]. Due to the large size of video content models, this can make such models difficult to interpret, use, and modify. For example, Duda *et al.*'s algebraic video [59] allows free form construction of models using semi structured annotations and hierarchic relationships. Such an approach could be used to create models resembling those of Rowe *et al.* and Sawhney *et al.* Moreover, as they share a common scheme, segments and their annotations from one model could easily be reused in another model. However, there is no description of the semantic and structural regularities of the models. Figure 17a illustrates this problem. In the figure, applications A and B share a simple scheme which states that any annotation (represented by white circles) can be related to any other by a relationship (represented by lines). Models in the two applications can therefore share annotations



a) Shared general scheme b) Fixed schemes

Figure 17: Simplified problem of reuse in current approaches

as necessary. However, unlike Figure 17b, there is no description of the semantic and structural regularities of the models *i.e.* there is no typing of annotations, and no specification of which types of annotations can be related.

3.2 What is Required to Address the Main Problem?

In order to address the problem of the lack of a common way of describing models (descriptions of particular interpretations of video content) and schemes (descriptions of the structural and semantic regularities of models) there needs to be some way of describing both schemes *and* models. This will result in schemes of one application being understood by another. Moreover, it will address the restricted reuse between models realised from different schemes.

Specifically, Chapter 2 identified the following characteristics of models that need to be described:

- **Segments.** These define a sequence video regarded as a logical unit. Considering video as a contiguous sequence of video frames they can be described by the index of the first and last frames of the sequence.
- **Annotations.** These represent an interpretation of the content of an associated segment. The annotation itself may be typed to reflect its content, and may contain different data such as free text, representative frame indices, structured data *etc.*
- **Relationships.** These define relationships between annotations. They can be relational (1-1: arbitrary or systematic) or inclusive (1-m or 1-x: taxonomy, arbitrary sequence, inferred sequence, specified sequence, or temporal).

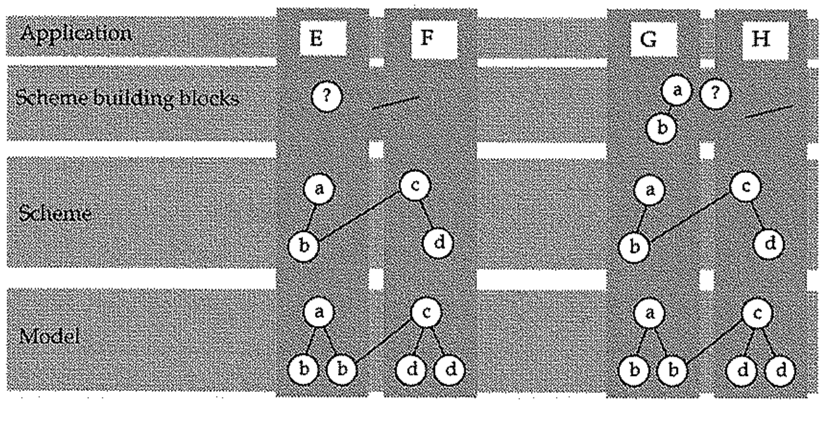
Schemes need to describe the semantic and structural regularities of classes of models (models sharing the same general structure and semantics); they need to describe what kinds of annotations there are, what information they store, and what kinds of relationships can exist between annotations.

An implicit requirement of Chapter 2 is that whatever is used to address this problem must be *accessible* to current and future users of video content models and schemes. This means that the approach should not require a large amount of effort on behalf of users to learn the concepts and use of the approach.

3.3 How Could the Main Problem be Addressed?

Currently reuse between models can be supported if the models share a common scheme. However, as discussed previously, this does not support the reuse of parts of models in models instantiated from different schemes. This thesis posits that describing schemes *and* models supports such reuse. The line of argument behind this position is: all models are realised from schemes. These schemes are themselves realised using a common set of scheme building blocks. Therefore one approach can use the building blocks to understand all the schemes created from it *and* the models realised from these schemes. Moreover, parts of schemes and models will be reusable in different schemes and models respectively. Furthermore, parts of models will be reusable in other models, including those created from different schemes.

Figure 18 illustrates this ability to reuse parts of models in models realised from different schemes (with respect to the illustration in Figure 17). In Figure 18a applications E and F share a common way of describing schemes; the scheme building blocks. In this simple example the building blocks state that annotations can have a type (represented by the question mark) and can be related in some way. These common building blocks mean that the applications can define their individual schemes and models, and parts of models can be reused in each others' models (indicated by the line between the two applications in the figure).



a) Aim of the thesis b) O-O frameworks

Figure 18: Solution to the main problem

This thesis is concerned with the definition and use of a *framework* which supports such description. In this thesis a framework provides the building blocks of schemes and models and supports their combination, refinement, and use (the top part of Figure 18a).

The term *framework* has a different meaning in the object-oriented programming field where it can mean 'a reusable design of all or part of a system that is represented by a set of abstract classes and the way their instances interact' [102] and/or 'a skeleton of an application that can be customized by an application developer' [102]. These definitions imply that some outline structures of schemes and models have already been defined and must be used in the development of schemes and models. For example, the provision of classes which support graphical user interface construction such as the awt (Abstract Window Toolkit) of Java [81]. By contrast, the framework in this thesis provides building blocks of schemes and models, but does not enforce any initial scheme or model to be developed from. This difference is reflected in Figure 18b which shows that the building blocks of object-oriented frameworks include an outline of a scheme to be used (annotations of type a can be related to annotations of type b). Whereas in Figure 18a there are no outline schemes enforced. This is beneficial in video content modelling as models are varied in structure and semantics; enforcing structures and semantics at this stage would be premature.

There are many approaches to describing the structural and semantic regularities of models (*abstract* modelling) as well as the models themselves (*concrete* modelling). For example, there are approaches in cognitive psychology which use *schemata* (see Eysenck *et al.* [60] for a discussion of schemata in cognitive psychology) to provide a framework for representing how generic knowledge may be stored and used by humans (abstract modelling). These schemata are then related to real world situations, for example to explain how predictions and inferences are made by people about such situations. However, this thesis considers approaches from computer science as they take into consideration the problems encountered when abstract and concrete modelling is supported by computers; video content models and schemes are to be supported by computers.

In computer science itself there are many approaches which support both abstract and concrete modelling. The three main approaches considered in this thesis originated in the 1970's from different parts of computer science: programming languages, knowledge representation for artificial intelligence, and database management. Formal approaches to abstract and concrete modelling such as the specification language Z (see Ince [97] for an introduction to Z) are not considered. This is because this thesis considers them to be inaccessible to typical users of video content models; one of the requirements of the approach outlined at the start of this chapter is that it must be accessible.

The following section discusses the three different computer science approaches in general terms. This is followed by discussion of the use of these approaches in hypermedia systems as these preceded video content modelling in the modelling of dynamic, temporal, complex media such as graphics, sound, and video. Finally, the use of these approaches for video content modelling is considered with respect to the strengths and weaknesses of the approaches in general.

3.3.1 Computer Science Approaches to Abstract and Concrete Modelling

3.3.1a Programming languages: object-oriented programming

In terms of programming languages, the concept of object-oriented programming was first developed in Simula [19], a simulation language developed in the 1970's. Developments in the 1980's, such as Smalltalk [79], supported this object-oriented approach for programming in general. Essentially, the object-oriented paradigm [143] represents real world entities by *objects* which have *attributes* and *behaviour* associated with them. In programming terms, these attributes are realised by variables, and the behaviour is realised by some programming code. Furthermore, objects may send messages to each other which may cause an object to perform some behaviour, and/or change the value of its attributes. This provides *encapsulation*; objects define a set of possible behaviours they can perform, but hide the manner in which it is performed. Another important characteristic of the paradigm is the provision for the definition of *object classes*. These define the default characteristics (attributes and behaviour) of the objects created from them. Object classes can themselves be used to define the characteristics of other classes by *inheritance*. A class which inherits from another inherits its attributes and behaviour, but may add to, or refine these characteristics. The concept of classes and inheritance supports one of the basic tenets of object-oriented programming: reuse [156][143]. A class defines the characteristics of all objects created from it and so the attributes and behaviour are reused. Similarly, inheritance allows classes to inherit, and so reuse, characteristics of other classes.

A very simple example of object-oriented programming follows: A class of objects could be defined to represent the behaviour and attributes of parking meters. Figure 19 illustrates such a simple parking meter. The attributes include: location, current parking time left, and total money collected. Behaviour includes: decreasing the time left, accepting money thus increasing the amount of parking time left and money collected, displaying the amount of time left, displaying a warning if insufficient time is left, and emptying the meter. Particular parking meters define their own location, total money inserted, and time remaining. Performing one of the behaviours involves either changing one of the attributes such as the amount of time left, or presenting some information to the user such as the current time left.

Object-oriented approaches typically only express one kind of relationship at the abstract level: the inheritance relationship *is_a*. As discussed previously, many different kinds of relationships such as hierarchic, temporal, spatial, need to be expressed at the abstract level for successful video content modelling. Furthermore, object-oriented approaches' relationships are *implicitly defined* by the processing or definition of objects. For instance, inheritance is defined by object classes rather than in some external, explicit

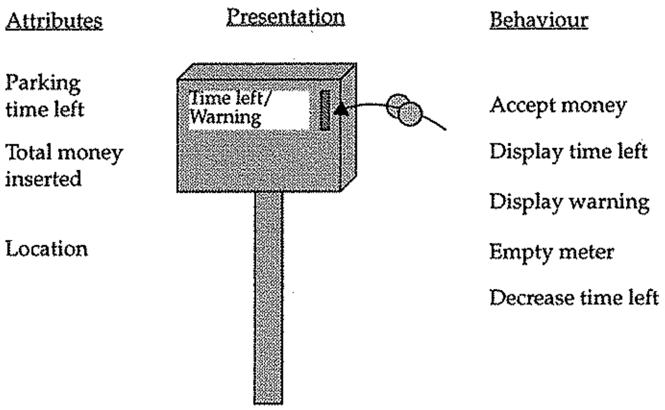


Figure 19: A simple parking meter

manner. Implicit definition of relationships makes them difficult to examine, query and manage [149]. Querying and examining relationships between segments of a video content model is identified in Chapter 2 as an important use of video content models.

3.3.1b Knowledge representation for artificial intelligence: frames

Artificial intelligence research tries to tackle the problems of representing large scale memory structures needed for reasoning about the world [69]. As such it is mostly concerned with representing declarative knowledge about the world in contrast to object-oriented programming approaches which attempt to model behaviour. The earliest of these approaches is Minsky's *frames* [138]. A *frame* is a representation of some aspect of the world *e.g.* an object or an event. A frame contains *slots* which are a particular piece of information; each slot has a *name* and a *value.* The value can be fixed or changeable, and may have a default value associated with it. There may also be constraints on the possible values of a slot. A frame can also describe general characteristics of aspects of the world *e.g.* types of objects or events. Specific descriptions are then generated by specifying that a frame *is_a* particular instance of another frame and specifying the appropriate information for the slots. Frames not only contain declarative knowledge, but can also associate *procedures* with slots. These can be used to make inferences about the appropriate values for the slot.

A very simple example of frame use follows. A frame could state in its slots that cats are of the feline genus, and they have a number of legs (normally 4), a breed, eyes which can be blue, green, or yellow, and a coat which can be white, grey, brown, or black. This provides a simplified description of certain physical characteristic of cats. A frame representing a particular cat states how many legs it has, which breed it is, the colour of its eyes, and whether the coat is white, grey, brown, or black. Procedures could be included in the eye colour slot to infer a cat's eye colour from its breed where possible *e.g.* Birman cats typically have blue eyes [204] so the eye colour slot should by default be set to blue if the breed slot is set to Birman.

In terms of relationships, frames are similar to the object-oriented approach discussed previously. Typically frames implicitly define *is_a* relationships to support inheritance. This leads to the lack of explicit descriptions of relationships, and the related problems discussed previously.

3.3.1c Database approaches: semantic data modelling

Hull *et al.* [96] provide a detailed history of semantic data modelling in which they highlight the original aim of semantic models which was to support the design of database schemas. Such schemas are used to structure databases; they define the records, their fields, and relationships between records. As semantic data modelling developed it was additionally used to help maintain database systems.

As with the other approaches considered in this chapter, semantic data models attempt to represent aspects of the real world. Typically this is done using *entities* related to each other by *relationships* (some early semantic data models were referred to as *entity relationship* models [36]). Entities are related to attributes which can take certain values and represent a particular aspect of the world. A schema describes the general properties being modelled and defines: entities, which entities are related to each other, and which attributes are associated with entities. Initially this appears to be similar to the notion of frames. Hull *et al.* discuss the main differences: frames typically only use *is_a* relationships whereas semantic models have much richer relationships, and frames can incorporate procedural knowledge making them more similar to object-oriented approaches than semantic data models. This distinction highlights the different modelling perspectives of semantic data modelling and object-oriented and frames approaches: semantic data modelling is interested in representing attributes of objects (declarative knowledge) whereas object-oriented approaches and frames are interested in modelling the behaviour and attributes of objects [96][111] (declarative and procedural knowledge). Video contains recordings of behaviour and so video content modelling is typically interested in modelling declarative knowledge (describing attributes of the content of video), not recreating the behaviour.

3.3.1d Synergy of object-oriented and semantic data models

Recent years have seen the convergence of object-oriented and semantic data modelling approaches [96]. This has typically taken the form of object-oriented approaches supporting the description for relationships between objects, and *visa-versa* the inclusion of behaviour in semantic data modelling approaches. However, these new approaches are typically just poorly supported extensions to the original paradigm. For example, object-oriented modelling and design approaches [167] provide an object-oriented design methodology which includes the notion of semantic relationships. However, the support for semantic relationships (including constraints and cardinality) is only explicit at the design stage. At the implementation stage these relationships are implemented using pointers or objects representing the relationships [167]. This means that at the implementation stage the relationships are implicitly defined. As discussed previously, implicit relationships are difficult to manage and query; querying relationships is an important use of video content models.

3.3.2 Hypermedia Approaches to Abstract and Concrete Modelling

First, a definition of hypermedia:

> "Hypermedia is a set of nodes of information (the "hyperbase") and a mechanism for moving among them" [150]

In this definition of hypermedia information can be any computer displayed media such as text, graphics, video, or audio. Furthermore, nodes may be composed of other nodes. Hypermedia subsumes the previous term *hypertext* in which information can only be static media such as text or still images. Typically the mechanism for moving among the nodes is the concept of a *(hyper)link* which consists of two connected nodes [159]. Therefore, moving around a hypermedia involves following links from one node to another. Links need not be restricted to connecting whole nodes together; *anchors* [82] may be defined within a node and linked to another anchor, or a node. An anchor is a subset of the node *e.g.* a word in a sentence, or a part of a graphic.

This thesis is concerned with the computer supported storage and use of video content models. Such video content models can be considered as a form of hypermedia: the video document is represented by a node which is composed of nodes representing annotations. These annotation nodes are themselves associated with segments of video *via* anchors. Finally, links between nodes represent relationships between annotations. Approaches such as Geißler's hypervideo [75] take this concept a step further by allowing anchors to be defined within frames of video.

With respect to hypermedia systems, concrete modelling involves the description of nodes, their content, and links between them. Abstract modelling involves the description of the content of classes of nodes (nodes having the same general structure and semantics) and the description of how nodes in a class are related to each other. Early approaches to hypermedia construction such as IDE [103] and Aquanet [132] only support static media such as text and still graphics. As hypermedia research developed, dynamic media such as moving images, sound, and video needed to be modelled, and more complex relationships needed to be expressed [67]. Generally, the motivation behind abstract and concrete modelling of hypermedia is to:

- Reduce development time by supporting the definition of semantic and structural regularities of applications [103].
- Encourage reuse of media between applications [16].
- Improve consistency and coherence of applications by regulating the structure of applications [132]. This is hoped to reduce user disorientation in the hypermedia

generated [34].

3.3.2a Generally applicable models of hypermedia

It is important at this point to distinguish between approaches which support the definition of abstract and concrete models, and approaches which provide generally applicable models of hypermedia. Approaches such as Puttress *et al.*'s Toolkit [159], the Andrew Toolkit [180], the Amsterdam Hypermedia Model [86], MHEG [158], Gaines *et al.*'s Open Architecture [67], Intermedia [206], Rheiner's Object description and representation [162], HyOctane and HyTime [112], the Dexter Model [82], and HyperBase [177] provide toolkits and/or models which support the generation of hypermedia applications from a predefined set of constructs. Such approaches do not support the definition of the structural and semantic regularities of classes applications. In terms of the previous discussion of problems with video content modelling, they are related to the approaches illustrated in Figure 17a (considering the circles to be nodes and the lines to be hyperlinks); they provide a general definition of hypermedia as nodes which can be linked to each other by hyperlinks.

Similarly, object-oriented approaches such as the Multimedia Component Kit [76][136] and STORM [3] support the subclassing of media objects to form new objects and classes, and support their presentation and synchronisation. Other approaches such as Kaindl *et al.*'s Structured Object Representation [106] and Tanaka *et al.*'s TextLink-III [189] use a frame based approach for structuring objects and so represent the semantics of classes of objects. However, these approaches do not model the structural regularities of classes of models; they only represent the inheritance hierarchy of objects to be used in the models. This essentially allows the creation of new building blocks for hypermedia applications, but does not support the specification of the structural and semantic regularities of classes of applications. Other approaches [84][122][90][157][160][30] concentrate on the presentation and synchronisation of dynamic media objects without any consideration for other relationships. This thesis is concerned with hypermedia approaches which do support abstract and concrete modelling.

3.3.2b Modelling hypermedia

The rest of this section highlights the interesting aspects of some approaches to abstract and concrete modelling of hypermedia with respect to the general approaches discussed in the previous section, and video content modelling.

The Instructional Design Environment (IDE) [103] was developed by Jordan *et al.* from NoteCards [83] primarily to support the construction of instructional hypertext. Of note are the system's *structure accelerators* which allow the definition of card (node) and link structure templates. A *template card* defines the text and still images that will appear in all cards created from it, and defines possible values for fields in such cards. Similarly, the *structure library* supports the definition of the types of links that should exist between cards when they are created. These concepts are similar to those of semantic data modelling; entities and their attributes, and relationships between them. Furthermore, where possible, IDE supports the automatic generation of links between cards from general links descriptions.

The related work of Marshal *et al.* [132] produced Aquanet from observations of the use of NoteCards [83], gIBIS [45], and Germ [25]. This tool attempts to support 'people trying to interpret information and organize their ideas' [132]. Aquanet models nodes of information (text in this case) using objects which are based on the notion of frames, and supports the definition of complex relationships between them. Object types define the slots of objects. Relationship types define relationships' names and the kinds of objects they can relate. These types (together referred to as schemas) are used to define the semantic and structural regularities of classes of models. Furthermore, types define how particular objects and relationships will be presented to the user on the screen.

Catlin *et al.*'s Hypermedia Templates [34] (developed from Intermedia [206]) uses *templates* to attempt to address the problem of user disorientation in hypermedia by reducing inconsistency in the hypermedia model. Their templates define the general structure of nodes in a class of hypermedia models; the content of nodes and the graphical layout. They also describe the types of links that can exist between nodes of hypermedia models. The main emphasis of the work is on defining hypermedia templates which can then be copied for a particular purpose in which the content of nodes is defined, and links between nodes are created. These templates define the structural and semantic characteristics of classes of hypermedia models.

Nanard *et al.*'s MacWeb [140][141] aims to tackle the same problem of user disorientation in hypermedia by supporting the definition of types of nodes and links. Again, such definitions are intended to support the development of consistent hypermedia models. In particular, MacWeb is aimed at supporting knowledge representation in hypermedia rather than simple hyperlinked graphs of information. The general concept is that *chunks* represent nodes and are defined from chunk types. Similarly, links relate chunks to each other and are defined from link types. This concept of node and link types supports the definition of structural and semantic regularities of classes of hypermedia models. In relation to the general approaches in the previous section, this approach is inspired by the object-oriented approach as chunks contain information and methods defined by chunk types which are similar to object classes. However, different types of links can be defined between chunks which is not typically supported by object-oriented approaches.

HB1 developed by Schnase *et al.* [175] uses a semantic data modelling paradigm coupled with the concept of objects from the object-oriented paradigm. Their aim is to provide more powerful structuring and connectivity of objects in hypermedia applications in order to supports users' perceptions of multimedia data rather than its physical representation. They model *applications* containing *components* which are the data manipulated by the application. They also model links and *associations* between components. Associations represent the structure of an application whereas links support hypermedia style traversal through the application. As they follow the semantic modelling paradigm the abstract description of applications, components, links, and associations are defined by schemas.

3.3.2c Modelling dynamic media

The hypermedia approaches discussed so far support the use of static media such as text and still images; they do not consider the temporal aspects of dynamic media such as audio and video. Simple temporal characteristics could be included in the descriptions of classes of models, but dynamic (temporal) media requires special support when being processed or displayed [90] and so such aspects must be a fundamental part of the approach. This is especially true of video which is a fundamentally temporal medium. The following approaches support the definition of structural and semantic regularities of classes of hypermedia models which consider the temporal aspects of dynamic media.

HyperStorM [16] provides an object-oriented approach to modelling the structural and semantic regularities of classes of hypermedia applications which use dynamic media. In HyperStorM objects represent multimedia data which can be static or temporal. They aim to help support the development of hypermedia applications and to encourage the reuse of media in different applications. Unlike other object-oriented approaches HyperStorM supports the definition of *semantic* links. Four basic link types are provided: flat hypermedia types for simple hyperlinks, element and set association types to represent the links in composite nodes, category specialisation types which support inheritance between objects, and role specialisation types which are used to describe the different roles objects can have in different contexts. These basic links types are represented by object classes and so can be subclassed to provide application specific links. Another interesting distinction between HyperStorM and other object-oriented approaches is that it separates the presentation of media from the object describing the media. This means that a piece of media such as video can be presented in different ways in different contexts. This supports their aim of reuse. Furthermore, HyperStorM supports user queries of models. The result of such a query is a set of objects which satisfy the query. This set is a new perspective on the model; the objects are not copied into the new set, but referenced using links. This reduces storage space requirements and preserves the consistency of the model. However, the separation of media objects from their presentation means that there is little support for temporal queries of models.

HDM (Hypertext Design Model) [71][70][73] supports the design and in part the implementation of hypermedia applications. The closely related HDM+ [35] addresses implementation of hypermedia applications in more detail. Both approaches are derived from the semantic data modelling paradigm using design based terminology.

Applications are authored in the *small* and in the *large*. Authoring in the large involves defining the structural and semantic regularities of a class of hypermedia applications; the product is a *schema*. Their motivations behind authoring in the large are: improvement of communication about the general characteristics of applications, development of design styles, reusability of general structures of application in new domains, providing consistent reading environments to reduce user disorientation, and supporting the development of design tools. A schema contains component and link types which define the structure and semantics of the class of applications. These types define the general characteristics and constraints on components and links generated from them. Authoring in the small, on the other hand, involves the creation of a particular application; an *instance* of a schema. In this case particular components (representing some multimedia data) and links are created from previously defined types. Three types of links are defined by HDM: perspective links to allow different perspectives of the same component, structural links which define hierarchic structures between components, and application links which are specific to the application and whose purposes are defined by its author.

An interesting aspect of HDM is its support for multiple perspectives of the same component *e.g.* switching between text and audio descriptions of the same component. Another interesting aspect of HDM is its ability to automatically generate links between components in a similar fashion to IDE. One of the most pertinent differences between HDM and other approaches is that HDM's components reference their multimedia data rather than storing it within themselves. This provides more flexible storage arrangements and allows different representations of the same basic data to easily be interrelated whilst consistency is maintained.

Early description of HDM did not include discussion of the temporal aspects of dynamic media although media such as audio and video was discussed. However, further developments such as HDM2 [72] did discuss the problems of temporal synchronisation of dynamic objects, especially with respect to user navigation through hypermedia applications.

Methodologies such as Isakowitz *et al.*'s RMM [98] (Relationship Management Methodology) and Schwabe *et al.*'s OOHDM [178] (Object-Oriented Hypermedia Design Model) use representations which have similar aims to HDM's. For example, RMM describes a methodology for the construction of hypermedia applications using a Relationship Management Data Model (RMDM) similar to HDM. Similarly, OOHDM addresses the problems of hypermedia application design methodology from the abstract to the concrete, but using an object-oriented approach.

3.3.3 Lack of Video Content Modelling Approaches

Hypermedia approaches which support dynamic media such as video initially appear to support video content modelling. However, the fundamental problem with using hypermedia approaches to support video content modelling is that they regard video as an atomic unit of information (a clip) [27]. Although some approaches such as HyperStorM [16] allow links and anchors to be defined within nodes such as a video clip, this does not meet the requirements of video content modelling set out in Section 3.2. First, it is not possible to use such approaches to define sub segments of clips. Second, only the clip itself can be annotated, not segments within the clip. Third, only simple hyperlinks rather than different kinds of relationships could be used within the clip to connect anchors. HDM [72] is an interesting exception as it is suggested that a video clip could be regarded as an ordered collection of nodes, each of which represents a video frame. This allows the definition of different collections of frames which could be regarded as segments. These could then be annotated and relationships created between them. However, this would make models of video enormous as typically 25 nodes per second of video would be needed. Furthermore, as HDM is aimed at the design of hypermedia applications and does not explicitly support the temporal characteristics of media such as video, it would be difficult to use it to support video content modelling.

There are other problems with using hypermedia approaches to support video content modelling. First, hypermedia approaches are concerned with data of relatively small size compared to the amount of data (video) required for video content modelling. Consequently, with the exception of HDM [72] and to a lesser extent HyperStorM [16] which allows access to external devices such as laser disc players, they tend to store information in the same node as its description. For video this becomes impractical due to the large amount of storage required. Furthermore, as video is then stored as clips within objects it leads to the previously discussed fundamental problem of using hypermedia systems to support video content modelling.

Another problem with hypermedia approaches is that the abstract description of a hypermedia application may include details of the graphical layout of nodes, such as their position on the screen [34][132]. This is inappropriate for video content modelling in which the same annotation may need to be presented in different ways in different contexts. For example, the different presentation and emphasis of topic annotations in ConText [53]. Moreover, with the exception of HyperStorM [16] and HDM [72], the responsibility for presenting objects' information lies with the objects themselves. This means that for video content modelling, an annotation (represented by an object) can only be presented in one way.

A final problem is that most hypermedia approaches which support dynamic media are object-oriented *e.g.* HyperStorm [16]. This means that fundamentally they are aimed at modelling behavioural and declarative knowledge. As stated previously, video content modelling is solely concerned with declarative knowledge. This makes such approaches less directly applicable than their semantic data modelling or frames counterparts *e.g* HDM [72].

As discussed in Section 3.1, there are no approaches to video content modelling which can describe the semantic and structural regularities of classes of video content models. Some approaches, such as Smoliar *et al.*'s [185] and Chua *et al.*'s [44] make use of frames but only to structure annotations and to construct topic categories to help users' classify and locate video content, not to describe the semantic and structural regularities of classes of models.

3.4 The Thesis

The are no approaches which currently directly address the problems outlined in Section 3.1. The question that remains is: what is the best approach to take to address these problems? This thesis posits that a semantic data modelling approach would be the most suitable because it possesses the following features:

1. Semantic data modelling supports descriptions of abstract and concrete models (schemes and models respectively).
2. Many different kinds of relationships can be described at both the abstract and concrete levels; video content models often require several kinds of relationships, not just composition and sequence relationships.
3. Relationships are explicitly represented (similar to *open* links used in systems such as Sun's Link Service [152] and MICROCOSM [54][63]). This makes them easy to examine and query [149][188] (identified in Chapter 2 as important uses of relationships), and able to support relationships between a large number of objects [54] which are typical in video content models. Furthermore, explicit relationships can be used to support different views of data [188] *i.e.* reuse of data. As discussed at the start of this chapter, reuse is a motivating factor for this thesis.
4. Objects represent declarative knowledge and not behaviour as object-oriented and frames approaches do. This is important because video content models contain declarative descriptions (annotations) of behaviour in video, they do not recreate the behaviour themselves.
5. Presentation and synchronisation is not the responsibility of objects; objects are presented and synchronised by external processes and so can be reused in different contexts in different ways. This is important for video content modelling as one segment and its associated annotation may be presented in different ways in different contexts; methods of presentation are implicit if they are the responsibility of the object.

To be concise, this thesis posits that: employing a semantic data modelling approach to video content modelling will allow the description of video content modelling schemes and models. Furthermore, such an approach will support different kinds of reuse of models' segments whilst retaining access to segments' original model and hence context.

The thesis as a whole demonstrates that such an approach can support a cross section of video content modelling through real world examples which cover a cross section of characteristics exhibited in Chapter 2. It demonstrates the viability of the semantic data modelling approach in a previously unexplored area: video content modelling. Furthermore, the work uses the examples to investigate the semantic data modelling based approach's support for different reuse of models' annotations and associated segments in different contexts, and in ways not previously supported.

It is important at this point to highlight the difference between the two aspects of this thesis: the demonstration of the suitability of the approach, and the investigation of the feasibility of its support for reuse. The demonstration makes stronger claims about the approach than the investigation. It claims that the examples will demonstrate the ability of the approach to describe video content models and schemes. The investigation, on the other hand, investigates the feasibility of the approach's support for different kinds of reuse in the domain of video content modelling. The reason for this distinction is that currently there has been little examination of support for reuse outside the software engineering field (see [114] for a review of software reuse). Consequently there is little understanding of the characteristics that need to be demonstrated to show support for reuse. Therefore an investigation of the feasibility of support for different kinds of reuse, and the problems raised needs to be undertaken before a demonstration can take place. Conversely, the demonstration of semantic data modelling's ability to describe abstract and concrete aspects of models has been undertaken in many different domains, including hypermedia [71][70][73][98]. This means that there is a better understanding of what needs to be demonstrated, and how it can be demonstrated. Therefore stronger claims can be made about the demonstration than the investigation.

This thesis is concerned with the computer support for video content modelling. As such it is part of hypermedia research; research which is concerned with the description and storage of multimedia information by computers, and relationships between such information.

3.5 Summary

This chapter discussed the main problem identified in Chapter 2; the lack of an approach which can describe video content models and schemes which results in the problem of restricted reuse of parts of video content models. Further, this chapter examined general approaches to abstract and concrete application design in order to provide some background to the problem addressed by this thesis. The chapter then considered the approaches used in hypermedia research which is closely related to video content modelling. Finally, this chapter highlighted the problem that there is no suitable approach in video content modelling or hypermedia, and proposed that semantic data modelling could be used to support video content modelling. This was then posited as the main aspect of the thesis. The next chapter provides a full definition of this thesis' semantic data modelling based framework to support video content modelling. It is followed by details of an implementation of such a framework, and then examples of the framework's use to demonstrate the thesis.

Chapter 4 VCMF Definition

This chapter describes the Video Content Modelling Framework (VCMF) which meets the requirements identified in Chapter 3 using a semantic data modelling approach. Chapter 6 supports this claim by giving examples of the framework's use. VCMF uses schemas defined in Section 4.3 to describe classes of video content models. These models are described by instances defined in Section 4.4, and can be queried using the query language defined in Section 4.5. Section 4.6 develops a graphical representation of VCMF which is used to illustrate the examples in Chapter 6.

4.1 Requirements Recap

To recap, Chapter 3 identified the following requirements of VCMF:

- It must be able to describe the semantic and structural regularities of video content models as well as describing models themselves.
- It must facilitate the reuse of parts of schemes, models, and associated video.
- It must be easily accessible to typical potential users of video content models.

The approach selected in Chapter 3 dictates that a semantic data modelling based framework should be used to satisfy these requirements.

VCMF is a schema based framework which uses terminology derived from Hull *et al.*'s General Semantic Model (GSM) [96]. Chapter 6 follows the demonstrative approach adopted in Garzotto *et al.*'s Hypertext Design Model (HDM) [70] of describing individual hypertext applications and classes of applications. Unlike HDM, VCMF is tuned to the particular problems of video content modelling identified in Chapter 3. These problems stem from the fact that video is a temporal media which is not usefully annotated as an atomic piece of hypermedia information. Furthermore, complex relationships between subsets of a video's annotated content need to be represented, and the reuse of these annotations and relationships should be supported to reduce the cost of using video in terms of storage space as well as collection, analysis, and annotation time [55].

4.2 VCMF Definition

Essentially, VCMF is a semantic model [96] which introduces the notion of time to objects, and allows constraints on relationships between objects to be described. As discussed in Chapter 3, it is not an object-oriented framework; objects do not perform any processing, they simply store information. Moreover, relationships (from [36]) between objects are explicitly modelled rather than being hidden within an object's processing as they are with object-oriented approaches [149].

Figure 20 illustrates how VCMF is used to describe schemes and models. Basically, in VCMF a *schema* describes a video content modelling scheme using *object types* which are related (linked in hypermedia terminology) to each other by *relationship types*. Similarly, an *instance* of a schema describes a video content model using *objects* (instances of object types) to describe segments and their annotations which are related to each other by *relationships* (instances of relationship types). Earlier versions of VCMF [27][28] used different terminology which was found to be confusing; the current terminology is intended to reflect the fact that a schema is a VCMF representation of a scheme and similarly an instance is a VCMF representation of a model. Schemes and models are less rigorously defined than schemas and instances. They are of most use in providing descriptions of work such as those presented in Chapter 2.

Object types define the *attributes*[1] that objects possess; relationship types define which relationships can exist between objects. These types are themselves instances of *base types* (object and relationship) which define the default (basic) attributes of object types and relationship types in VCMF. This is essentially a simple form of inheritance.

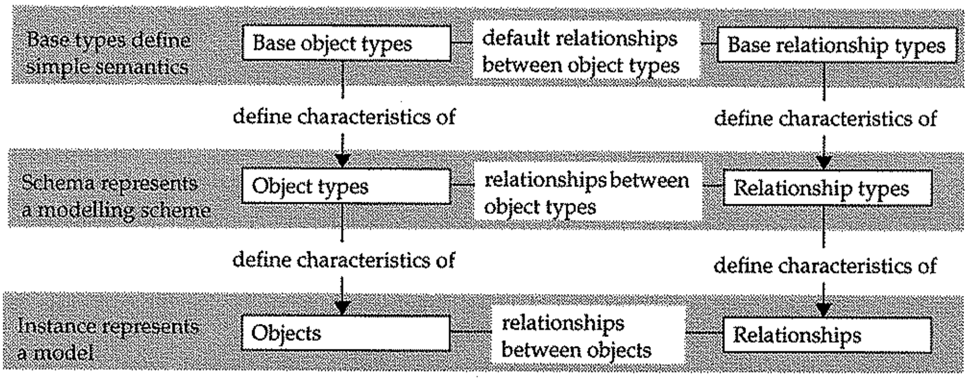

Base types define simple semantics
Base object types
default relationships between object types
Base relationship types
define characteristics of
define characteristics of
Schema represents a modelling scheme
Object types
relationships between object types
Relationship types
define characteristics of
define characteristics of
Instance represents a model
Objects
relationships between objects
Relationships

Figure 20: Relationship between schema and instance

In VCMF frames of video are assumed to be uniquely indexed by a frame number[2]. The frames are also assumed to be stored in an ordered sequence in a randomly accessible storage medium (for example, a video disc [126]). A segment (a linear sequence of frames) can therefore be defined by its start and end frame numbers, referred to as its *temporal extent*. As discussed in Chapter 2, this approach is used by current uses of video. Also discussed in that chapter is the fact that segments can share frames of video; they can overlap unlike clips which cannot. This is ideal for most uses of video discussed in Chapter 2. However, as discussed in Chapter 8, it does mean that if the notion of clips is important to the use of video, it cannot be enforced in VCMF.

[1] In VCMF attributes are name and value pairs. The name is textual whereas the value is the product of BNF rules (see Section 4.3.1). Multimedia data are used as attribute values in some video content modelling systems *e.g.* some of the systems identified in Chapter 2 have audio attributes. For simplicity, these kinds of data are not considered in this thesis.

[2] Some compressed video storage formats such as MPEG [68] do not store individual frames, but instead store changes to certain frames. This thesis assumes that there is some mechanism which can compute and present individual frames from such a format.

A brief example of VCMF terminology[1] follows. VCMF represents the News Analysis scheme in Chapter 2 as a schema. Object types are used to describe the different types of annotations in the example scheme: News programme, Title shot, Weather shot, News report, Anchor person sequence, News reel sequence, Anchor person shot, and News reel shot. Relationship types relate these object types to each other. For instance, a News programme object type is related to the Title shot, News reports, and Weather shot object types by composed_of relationships. Furthermore, sequential relationships relate these object types to each other to indicate that a Title shot is followed by News reports, which are followed by a Weather shot. The composed_of and sequential relationship types are the most common relationship types found in video content modelling. More unusual relationship types (*e.g.* spatial) are illustrated in Chapter 6 and highlight the flexibility of VCMF's approach.

Continuing the example, in VCMF a model of a particular news programme is represented by an instance of the schema. Each object represents an annotation and its segment, and is instantiated from an object type in the schema. For example, there is an object of object type Anchor person shot whose textual description is "Introduction". This object is associated with the appropriate segment of video. Two kinds of relationships are used in the instance: composed_of and sequence. A composed_of relationship relates objects to objects at a lower level of the hierarchy. For example, the News programme object is related to the Title shot object, News report object, and Weather shot object by composed_of relationships. sequence relationships represent the linear sequence of objects. In the model they are used to represent the fact that the Title shot object comes before the News report object which comes before the Weather shot object. A full description of this simple example represented using VCMF is given in Section 4.6.

The detailed definition of VCMF and its graphical representations used in Chapter 6, are presented in the following sections.

4.3 Schema

A schema describes the structural and semantic regularities of instances created from it. It does this using object types to describe the semantics of objects in an instance of the schema, and relationship types to describe which relationships may exist between these objects. A schema itself may be reused to create different instances, and types may be reused in new schemas.

4.3.1 Object types

Object types define what information is stored in objects and may determine certain

[1] To make it clear which information relates to the contents of VCMF base types, schemas, and instances, all such information is presented in Helvetica. Such information includes names of: objects, relationships, types, attributes and their values. However, names of base types always start with a capital letter but are not presented in Helvetica.

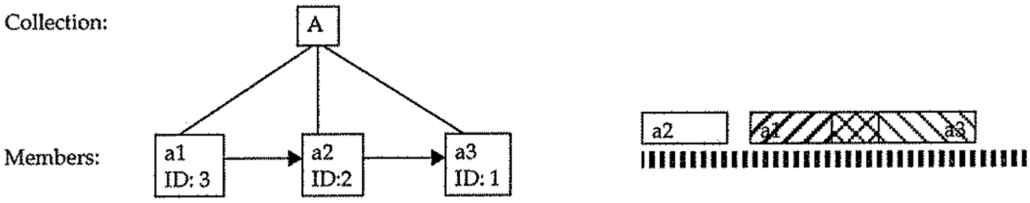
relationship types that are used for Collections (see later discussions). Each object type has a unique object type identifier (TID) which is not typically displayed to the user but is present because object types may not be uniquely identified by their attributes (as discussed by Hull *et al.* [96]). For the same reason relationship types have TIDs and objects and relationships have unique identifiers (IDs). For simplicity, object types are often referred to by their name attribute. This convention supports the need for typed annotations discussed in Chapter 2 and Chapter 3.

Object types themselves are created from base object types which define their attributes and basic semantics. VCMF defines two main types of base object types: *Units* and *Collections*. In an instance, Unit based objects (Units for short) are regarded as atomic units of information whereas Collection based objects (Collections for short) are composed (using relationships) of other (*member*) objects which may themselves be other Collections or Units. Units always have a temporal extent as they directly relate to a segment of video whereas Collections need not have a temporal extent as they may refer to many disjoint segments. This inclusion of a temporal extent, and hence relating the objects to linear sequences of video frames is a significant aspect of VCMF. Other semantic data models do not provide this explicit link between objects and a temporal media such as video. It is essential for the efficient modelling and use of video that this explicit link exists. If it did not exist, there would be problems of maintaining consistency between the model and the video that it represents.

4.3.1a Collections

In an instance a Collection is related to its member objects using relationships specified by its object type. Similarly, its member objects are related to each other by relationships specified by its object type. Figure 21a illustrates a simple example of a Collection A, its member objects a1, a2, a3, each with an ID attribute (returned to in later sections), and the relationships between them. The lines from A indicate that a1, a2, and a3 are members of A. The arrowed lines indicate the sequence of the member objects: a1 → a2 → a3. The member objects are not necessarily stored in this linear sequence in the storage device, and they may have frames in common. Figure 21b shows a possible storage of the three member objects a1, a2, and a3. The stripes at the bottom of the figure represent the linear sequence of frames; the rectangles above indicate which frames are associated with which objects. In this example the member objects are not stored in the same sequence as Collection A orders them, they are not temporally abutted, and two of them overlap in terms of the frames they contain (indicated by the cross-hatched area).

The rationale behind this method of representing collections of objects is that VCMF is concerned with semantic data modelling of video content rather than an object-oriented modelling approach such as STORM [3]. In semantic data modelling, relationships are as important as objects, and all relationships between objects are explicit. In object-oriented approaches, relationships such as membership and sequence



a) Collection A and its member objects

b) Possible storage of members

Figure 21: Simple example of a Collection, its members, their relationships, and possible storage

are implicitly calculated by an object's processing [149]. Explicitly storing relationships means that they can more easily be queried and navigated [149][188] in object independent ways. A full discussion of the rationale for basing VCMF on a semantic data modelling approach is given in Chapter 3.

4.3.1b Base collection types

VCMF defines five base types of Collections which differ in how member objects are related to each other. The relationships between a collection and its members, and between members of a collection, are created and maintained by software which supports VCMF (*e.g.* the toolset discussed in Chapter 5) rather than being implicitly defined by object processing as they are in object-oriented approaches. The five base types of Collections are:

1. **Un-Ordered Collection**: In an Un-Ordered Collection there are no relationships between member objects. These are typically used to create indices of objects. For example, Rowe *et. al.*'s Video Database Browser [165] collects available videos into an index which would be represented in VCMF using an Un-Ordered Collection. Although the index of videos may be presented in an order, there is no ordering amongst the videos themselves. That is to say, the meaning of an Un-Ordered Collection is *a collection whose member objects may be presented in a sequence* rather than an Ordered Collection whose meaning is *a collection whose member objects (videos in this example) are presented in a sequence.*
2. **Ordered Collections**: There are three base Collection types in which member objects are ordered in some sequence (which may be circular in which case the member object following the last member object is implicitly the first member object). The first member object in the linear sequence within an Ordered Collection can be identified as it has no other member objects of that Collection preceding it. Similarly, the last member object of the sequence precedes no other member objects of that Collection. Member objects are ordered within a Collection according to the following sub types:

   2a. **Attribute-Ordered Collection**: the member objects are ordered in terms of an attribute. For example, if Collection A of figure Figure 21a were an Attribute-Ordered Collection which ordered its members on their ID attribute then the sequence of member objects would be a3 (ID: 1) → a2 (ID: 2) → a1 (ID: 3) rather than a1 → a2 → a3 as illustrated in the figure. Such Collections are typically used to provide alphabetical listings of objects (for example using a Name attribute), or to order the results of queries according to their relevance.

2b. **User-Ordered Collection**: no decision about the ordering of member objects is made until run time when the user either selects an attribute to order them with, or manually orders them. This is typically used to facilitate flexible viewing of query results, for example, viewing the results of queries from Rowe *et al.*'s Video Database Browser [165]. If Collection A of figure Figure 21a were a User-Ordered Collection, the user may impose any sequence on the objects at run time *e.g.* a3 → a1 → a2 rather than a1 → a2 → a3 as illustrated in the figure.

2c. **Pre-Ordered Collection**: the member objects are ordered according to a sequence which is defined in the schema using relationship types. This is the only Collection which explicitly states the ordering of its member objects; all other Collections infer any ordering from attributes of the member objects. This type of Collection is typically used when an analyst wishes to explicitly state the ordering of member objects, and then extract these member objects from the video source. For example, if Collection A of figure Figure 21a were a Pre-Ordered Collection which stated that member objects must be ordered such that objects a1 always precede objects a2, and objects a2 always precede objects a3 then the sequence of member objects illustrated in the figure would be permitted. Such collections are appropriate for the news analysis example considered in Chapter 2. That example states that a News programme's first member object is a Title shot which is followed by some News reports which are in turn followed by one Weather shot; Pre-Ordered Collections support such explicit statements about the ordering of their member objects.

3. **Temporal Collection**: In a Temporal Collection the member objects are related to each other by Temporal relationships which describe their relative temporal positions in the underlying sequence of video frames (see Section 4.3.2 for a description of Temporal relationships). They are different to Ordered Collections which relate member objects in a continuous sequence; Temporal Collections' members may temporally overlap and there is not necessarily a continuous sequence of members. These Collections are typically used when examining the relative temporal positions of member objects. Hibino *et al.* [91] describe a sample scenario in which the temporal relationships of a video content model are examined to discover trends in social interactions between children and teachers in the classroom. Figure 21b illustrates the temporal nature of member objects of Collection A in Figure 21a. If Collection A were a Temporal collection then relationships would be created between the member objects to indicate their Temporal relationships: a2 Precedes a1, a2 Precedes a3, and

a1 Overlaps a3.

With the exception of Un-Ordered Collections, Collections can have a temporal extent. As discussed previously, frames of video are assumed to be stored in a linear sequence in a randomly accessible medium. The segment of video associated with an object is defined by the object's temporal extent (the start and end frame number of the linear sequence of frames that constitute the segment). This can either be derived from the member objects by taking the smallest start frame number as its start frame and the largest end frame number as its end frame (*e.g.* for Collection A in Figure 21, the start_frame of a2 and the end_frame of a3), or it can be specified by the user. If the temporal extent is specified by the user then it must be *greater* than the derived temporal extent *i.e.* the start frame number must be less than the derived start frame and the end frame number must be larger than the derived end frame. Deriving the temporal extent of a Collection may mean that the Collection is associated with frames that none of its members are. Collection A in Figure 21 illustrates this. If A temporally bounds a1, a2, and a3 then it is associated with frames that are between a2 and a1 (illustrated in Figure 21b).

The different kinds of Collections in VCMF reflect the inclusive relationships discussed in Chapter 2. Table 6 shows the match between kinds of Collections and inclusive relationships.

Collection	Inclusive relationship
Un-Ordered	Taxonomy
Attribute-Ordered	Inferred sequence
User-Ordered	Arbitrary sequence
Pre-Ordered	Specified sequence
Temporal	Temporal

Table 6: Match between kinds of Collections and inclusive relationships

4.3.1c Object type attributes

Each object type may define a list of *unique* attribute names and value types. These store information which is particular to the object type (*i.e.* not defined in the base object type); the value type defines Backus-Naur Form rules used to generate the attribute value in the instance. This use of attributes and BNF rules supports the structuring of annotations of segments discussed in Chapter 2.

The BNF rules defined for VCMF are presented in Table 7. A simple example of their use is an object type which has a unique attribute value type of Shot <number>.<lower-letter>. In an object instantiated from this type the unique attribute could have the value Shot 2.a, but not Shot x.A.

In addition to the rules presented in Table 7 VCMF supports the definition of *derived*

<digit> ::= 0|1|2|3|4|5|6|7|8|9
<number> ::= <digit>|<digit><number>
<sign> ::= +|-
<signed-number> ::= <sign><number>
<lower-letter> ::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
<upper-letter> ::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
<letter> ::= <lower-letter>|<upper-letter>
<string> ::= <letter>|<letter><string>
<alpha-numeric> ::= <digit>|<letter>
<punctuation> ::= !|,|.|?|;|:|’|”
<symbol> ::= ~|<|>|/|[|]|{|}|@|£|$|%|^|&|*|(|)|-|_|+|=
<character> ::= <alpha-numeric>|<punctuation>|<symbol>
<text> ::= <character>|<character><text>
<date> ::= <digit><digit>/<digit><digit>/<digit><digit><digit><digit>
<time> ::= <digit><digit>:<digit><digit>.<digit><digit>

Table 7: BNF rules defined for VCMF

attributes [96]. In VCMF, these are supported by the value rule which evaluates, in the instance, to the value of the attribute in the object of type object type related to this object by a relationship[1] of type relationship type:

<value(relationship type, object type, attribute)>

For example, the unique attribute value type Shot <value(parent_of, Scene, Identifier)>.<lower-letter> uses the value rule. In the instance, this would take the value of the Identifier attribute of the object of type Scene related to the object by a relationship of type parent_of.

In total, each object type has the following attributes:

- A unique identifier (TID).
- Its name.
- The base object type that it is based upon (supporting simple inheritance).
- If appropriate, the relationship type(s) which relates it to its member objects.
- If appropriate, the relationship type(s) which relates its member objects to each other.
- If appropriate, whether it is a circular Ordered Collection.
- Whether the instance will have a temporal extent, and if so, whether it is derived or

[1] Note that if more than one object of type object type is related to the object by relationships of type relationship type, then the first object related is considered.

user defined.

- A list of unique attribute types.

4.3.2 Relationship types

Relationship types describe the relationships that are applicable between objects in an instance of a schema *i.e.* they describe the structural regularities of instances created from the schema. They describe the relationships between Collections and their members, between members of Collections, and other non Collection oriented relationships not so far discussed.

4.3.2a Relationship constraints

A relationship type imposes three kinds of constraint on relationships instantiated from it. First, a relationship type specifies which types of object the relationship can be instantiated between. Relationships relate two objects referred to as the *source* and *destination* objects [73]. This first constraint is achieved by storing the TID of the source and destination object types, and only allowing a relationship to be instantiated between objects with types of the appropriate TIDs.

Second, the *cardinality* [109] (discussed in Chapter 2 and Chapter 3; whether the relationship is one-to-one, one-to-many, *etc.*) is specified by the relationship type. This is achieved by storing a *cardinality list* for the destination object (there is always assumed to be one source object). A cardinality list constrains how many relationships may be made to destination objects. It contains either:

- A list of integers ≥0 which indicate the possible number of destination objects. A list containing 0 indicates that the relationship need not exist. Examples of relationships and their cardinality lists (in curly brackets) are: one-to-one {1}, one-to-one or one-to-two {1,2}, one-to-one or one-to-two or one-to-three {1-3}.
- The letter m which indicates that there should be more than zero destination objects.

Third, the relationship type may optionally specify constraints on the source and destination objects' attribute values. These constraints are not often used by viewers of models. They are more of interest to authors of models as they help to define the semantics of the relationship types by defining how the relationship types can be used. The Temporal relationships discussed later in this section are a pertinent example of the use of these constraints to define the semantics of relationships.

Constraints are supported by storing a *constraint specification* expressed using predicate calculus; if it resolves to true then a relationship of that type may be instantiated. A constraint specification prefixes attribute names of the source object with s. and the destination object with d.

For example, a schema whose object types represent points (x, y) on a plane (they have x and y attributes) may wish to state that the relationship called is_due_north_of only holds in the instance when the source object's x attribute is the same as the destination object's x attribute and the source object's y attribute is greater than the destination object's y attribute (considering north to be in the positive y direction). This is expressed as:

$(s.x = d.x) \wedge (s.y > d.y)$

4.3.2b Relationship names

Relationship types have a unique identifier and one or two names: its normal name (from source to destination) and optionally its inverse name for when the relationship is viewed from destination to source. Relationship types with no inverse name are considered to be unidirectional as opposed to the default of bidirectional.

In VCMF, the naming convention for relationship types is that they should be in *verb* form. Given a relationship r with normal name n between source object with name s, and destination object with name d, the sentence 's n d' should be understandable. For example, 's is_parent_of d'[1], 's is_after d', 's meets d'. The inverse name i follows the same convention *i.e.* 'd i s' should be comprehensible.

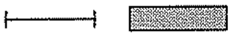
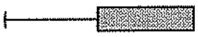
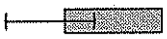
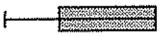
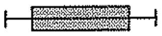
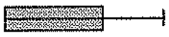
4.3.2c Base relationship types

In keeping with object types, relationships types are based on base relationship types. VCMF defines four base relationship types which provide the default names, cardinality lists, and constraint specification of relationship types without the source or destination TIDs. Relationship types based on these base types can change the name and cardinality list; the base types are used to provide the underlying semantics of the relationship type. In the following points the base relationship types are named in bold, followed by a definition of the default characteristics of relationship types defined from them including the default names, constraints, and cardinality where appropriate:

1. **Membership:** Default name: Contains, default inverse name: Is_a_member_of. A 1-m relationship type used to describe the relationships between a Collection and its members: a Collection *contains* ≥ 0 members.
2. **Sequential:** Default name: Precedes, default inverse name: Follows. A 1-1 relationship type used to describe the linear ordering of objects, typically members within an Ordered Collection: the first member of an Ordered Collection *precedes* the second member. This form of sequence relationship is used in VCMF as it is the most common form of sequence relationship found in video content modelling. Other forms of sequence relationship can be defined as necessary using Schema specific relationships (discussed later).
3. **Temporal:** a group of relationships typically used in Temporal Collections to describe the relative temporal relationships between instance objects. These relationships are based on the thirteen primitive temporal relationships described by Venema [200]

[1] Underscores often replace spaces in relationship type names to indicate that the whole phrase is a relationship name.

and Allen [6]: *before, meets, during, starts, finishes, overlaps,* their symmetric counterparts, and *equals.* Table 8 defines Temporal relationships' normal and inverse names, and their constraint specification with reference to the graphical notation illustrated in Figure 22. In the graphical notation time is regarded as increasing from left to right. Two instance objects A and B are illustrated; the start_frame of A is a_0, B's is b_0, the end_frame of A is a_1, B's is b_1. It is assumed that $a_0 < a_1$ and $b_0 < b_1$. Also, as mentioned previously, frames of video are assumed to be numbered sequentially with each frame being uniquely indexed. When objects are added to a Temporal Collection the appropriate Temporal relationships between the objects are automatically set up and maintained by the VCMF support system. This allows the user to easily view, navigate, and compare the Temporal relationships that exist between the objects.

Graphical representation	**Normal name**	**Inverse name**	**Constraint specification**
	Precedes	Follows	s.end_frame < d.start_frame
	Meets	Is_met_by	s.end_frame = d.start_frame
	Overlaps	Is_overlapped_by	s.start_frame < d.start_frame ∧ s.end_frame > d.start_frame ∧ s.end_frame < d.end_frame
	Finishes	Is_finished_by	s.start_frame < d.start_frame ∧ s.start_frame = d.start_frame
	Contains	Is_during	s.start_frame < d.start_frame ∧ s.end_frame > d.end_frame
	Starts	Is_started_by	s.start_frame = d.start_frame ∧ s.end_frame > d.end_frame

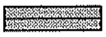
	Equals	Equals	s.start_frame = d.start_frame ∧
			s.end_frame = d.end_frame

Table 8: Temporal relationships

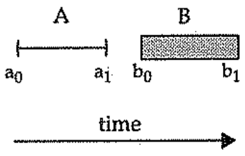


Figure 22: Graphical notation for Table 8

4. **Schema specific**: these types have no default values for names, constraints or cardinality; these must be defined by the schema. These relationship types allow the

schema to specify relationships which do not fit into the base relationship types previously detailed. They represent the arbitrary and systematic relational relationships discussed in the taxonomy of relationships of Chapter 2. Systematic relationships are distinguished from arbitrary relationships as they use a constraint specification to define possible destinations, so enforcing some systematic rules on the destination. For example, a relationship type references which can only lead to objects with certain attribute values might be needed in the schema. This is neither a Membership, Sequential, or Temporal base relationship type and so is based upon the Schema specific base type.

As discussed previously in this chapter, and in Chapter 2, and Chapter 3, relationships involved in Collections (Membership, Sequential and Temporal relationship base types) are explicitly represented in VCMF. To reiterate, this is because VCMF takes a semantic data modelling approach in which all relationships are explicit rather than some being implicitly generated by objects in object-oriented approaches.

4.3.2d Relationship type attributes

In summary, a relationship type has the following attributes:

- A unique identifier (TID).
- Its base relationship type.
- Normal and optional inverse textual names.
- The TID of the object type that this leads from (the source).
- The TID of the object type that this leads to (the destination).
- A destination cardinality list.
- An optional constraint specification.

The names and base relationship type of a relationship type give it a simple semantic. The cardinality list and constraint specification ensure that this simple semantic is used consistently in instances.

4.3.2e Advantages of the approach

Garzotto *et al.* [71] describe three advantages of their schema based approach to hypermedia design: the ability to check for completeness, the consistent use of relationships, and the automatic generation of relationships. VCMF shares these three advantages as follows:

1. An instance which does not have all the relationships deemed necessary by the schema is *incomplete*. This is an error in the instance's construction as objects may not be appropriately related to each other. Instances will often contain hundreds of objects and relationships and so the framework's ability to check whether all appropriate relationships have been instantiated, and if not, which ones are missing, is a great assistance determining the completeness of instances. Chapter 5 discusses the currently implemented approach to supporting such checks.

2. The constraining mechanisms applied to relationships in VCMF ensure that they can only be instantiated between appropriate objects *i.e.* the constraints of the schema are carefully adhered to in the instance by VCMF. As Garzotto *et al.* discuss, this is beneficial as it ensures consistent use of relationships and encourages the schema creator to carefully describe relationships rather than using simple unnamed *hyperlinks* which do not reflect the semantics of their relationship [56][146].
3. Relationships between member objects of Collections are automatically instantiated and maintained. Furthermore, Schema specific relationships are also automatically instantiated and maintained where possible. Garzotto *et al.* claim that this reduces the authoring effort of the user which could otherwise be overwhelming due to the large size of instances.

4.4 Instances

An instance represents a structured description of an interpretation of the content of a video recording. It does this using objects to describe both the segments and their annotations. Objects are related to each other by relationships. The semantics and structure of an instance is determined by the schema that it is an instance of. An instance itself may be reused within other instances, but more frequently objects are reused in different instances by including them in multiple Collections, or by using other relationships to include them in multiple instances. This form of reuse reduces the time spent creating instances as some of the objects are already defined, and prevents multiple copies of the same video recording being used, therefore reducing storage space and promoting consistent annotation of segments. It also provides the ability to easily view different uses, or interpretations, of the same video footage.

4.4.1 Objects

An object represents an annotation and possibly an associated segment of video.

The object type used to instantiate an object determines whether it is associated with a video segment, whether it is a Collection or Unit (and so atomic), and what unique attributes are required. The first unique attribute of an object is regarded as its *key* attribute which is the most important description of the object with respect to the object type's name. For instance, a News programme in the news analysis example has several unique attributes including the channel and the date and time of broadcast, but the key attribute is its title. The value of a unique attribute must be constructed from the BNF rules specified in the unique attribute value type in the object type.

Each object has attributes for:

- A unique identifier (ID).
- The name of the object type.
- The TID of the object type that it is an instance of.
- A piece of representative information.

- Temporal attributes (if prescribed by the object type): start_frame and end_frame attributes which are the start and end frame numbers of the video segment represented.
- Unique attributes (if any).

4.4.2 Relationships

Relationships represent the relationships between annotations and in doing so provide navigation paths between objects of an instance. Relationship types determine which objects can be related, the cardinality of the relationship, its name(s), and whether it is unidirectional or bidirectional. A relationship has the following attributes (its name(s) *etc.* can be found by examining the relationship type it is instantiated from):

- A unique identifier (ID).
- Source object ID.
- Destination object ID.
- The TID of the relationship type that it is an instance of.

4.5 Querying an Instance

The ability to query an instance provides a fundamental form of reuse. In VCMF this reuse is strengthened by taking the approach of other research such as VideoSTAR [93] and HDM [35], and making the results of queries persistent within the instance rather than transient. This supports both iterative refinement of queries, and the reuse of query results at a later date. This persistence is achieved by storing the result of a query in a Collection. Typically this Collection is a User-Ordered Collection which allows it to be ordered by the user when they submit the query.

A query language for VCMF needs to:

1. Support queries of all object attributes (including temporal extents).
2. Be simple and accessible. VCMF is not primarily concerned with queries and query languages; the query language is used to show one form of reuse, it should not dominate the framework or the thesis.

4.5.1 SVQL: Simple Video Query Language

Video database systems such as those described by Rowe *et al.* [165] support queries of instances. They typically use query languages based on SQL [115]. VCMF defines a subset of SQL called SVQL (Simple Video Query Language). This meets the two requirements above; it is simple (supporting one kind of query, the select), and it supports querying of all kinds of attributes. SVQL's select is defined as:

select from *object types* into *object type* where *expression* [order by *attribute name*]

The emphasised text indicates the kind of information that should be present in the query as follows:

- *object types* defines a list of the types of objects that will be queried.

- *object type* defines the object type that will be used to instantiate the Collection to hold the result of the query.
- *expression* defines what is to be searched for; it can contain object attribute names (*e.g.* start_frame), comparisons (≤, <, =, >, ≥, ~=[1]), and logical operators (AND, OR, NOT).
- *attribute name* defines the attribute that is used to order the member objects of the Collection created as a result of the query. This part of the query is optional, denoted by its enclosure square brackets.

The result of a query is a Collection containing all objects for which the query expression is true, and whose type is as specified. This Collection may be ordered according to the query.

The first example query in Rowe *et al.*'s [165] discussion of querying their video database is: 'movies produced by Universal in the 1960's in which Elvis did not appear'. Assuming that a schema contained object types named Movie (an Un-Ordered Collection with unique attributes Organisation, Date, and Actors) and Query (a User-Ordered Collection), this query could be specified in SVQL as:

```
select from Movie into Query where (Organisation = "Universal") AND (Date ≥ 1960) AND
(Date < 1970) AND (NOT Actors ~= "Elvis")
```

Note that for ease of reading this example query uses object type names rather than their TIDs.

The result of submitting this query would be the creation of an Collection of type Query which contains the Movie objects that match the query expression.

4.6 Graphical representation

This section describes the graphical representation of schemas and instances used in the examples of Chapter 6. To improve the clarity of diagrams this representation only shows the names and unique attributes of schemas and instances.

There are many, many graphical representations of models in use, let alone graphical representations of semantic data models [196]. The graphical representation presented in this section is yet another such representation. It has the following aims:

1. To provide a concise representation of schemes and models.
2. To distinguish between atomic and non-atomic objects and object types.
3. To make the representation of models similar to their scheme.

The rationale behind defining another graphical representation is presented in the following points:

1. Unlike other semantic data model representations, VCMF attributes are considered part of the object rather than being associated with an object. To this end they are included in the graphical representation of the object rather than being separate (similar to graphical representations of Minsky's frames [138]).

[1] ~= is the *contains* comparison. a ~= b is true if a contains b.

2. VCMF places great importance on the definition of relationships between member objects, and between member objects and their Collection. Other semantic data model representations represent aggregation, but do not provide explicit ways to define the relationship types that will exist between member objects of a model.
3. VCMF models tend to contain hundreds of objects and relationships. Representing these models using other representations may be visually overwhelming as relationship names are attached to each relationship and attributes are considered to be external to the objects (as discussed previously). Typically in these representations only the *is_a* relationship is represented by a special type of line without associated text.

To meet the above aims, and supported by the rationale, this graphical representation has the following basic characteristics:

1. Objects and object types are represented by boxes which are either thick lined to indicate an atomic object, or thin lined to indicate a collection.
2. Attributes are listed in the object or object type box.
3. Relationships are usually defined using a key so that names do not need to be appended to lines. This is aimed at reducing clutter in the diagram.
4. Relationship types to be used between member objects are explicitly represented in schemas. Furthermore, these intersect the Membership relationship types to indicate which member object types are to be related. This is inspired by Diskin *et al.*'s *sketches* [58] which uses arcs to indicate grouping of relationships. The addition here is that arcs are also used to indicate the relationships that will exist between member objects, and possibly also the order of member objects.

4.6.1 Graphical representation of schemas

Figure 23 illustrates an example use of the schema graphical notation presented in Table 10. This schema describes the News Analysis scheme[1] discussed in Chapter 2. The boxes represent object types and their attributes, the straight arrowed lines represent relationship types and the curved lines represent the types of Collections.

4.6.1a Graphical representation of object types

Considering the object types first, the thick edged boxes represent object types based on Units whereas thin edged boxes represent object types based on Collections. The bold text contained in the boxed is the object type's name. This is followed by its unique attributes; these are organised as the attribute name followed by its value type underlined. In the instance the underlined text will be instantiated with a value of the

[1] It must be emphasised that the News Analysis example of Chapter 2 is very simple and artificial. Its purpose is to help explain various video content modelling concepts. It is used here to help illustrate the use of the graphical notation which is used in more complex examples in later chapters. Moreover, the use of text attributes is a simplification; such applications typically also use multimedia data such a still images or audio for attributes.

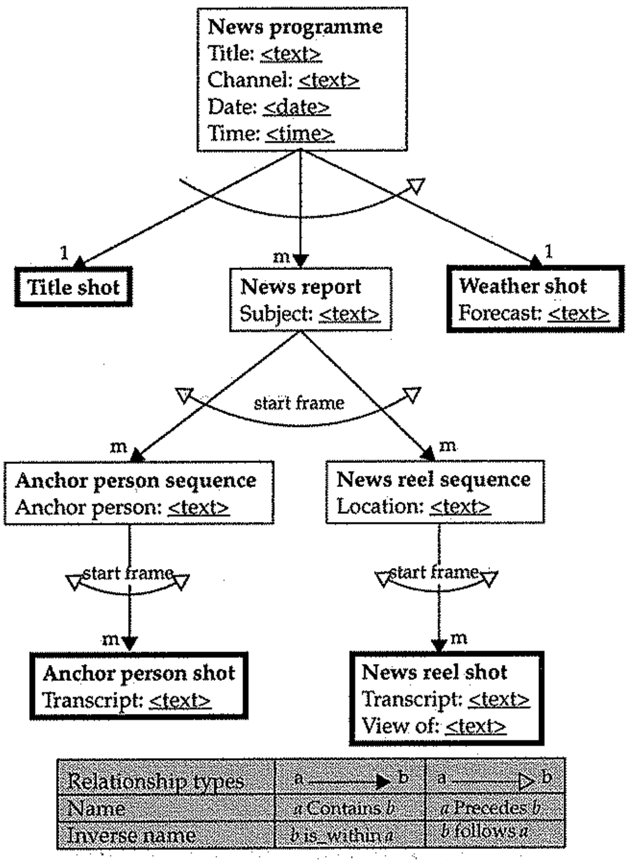


Figure 23: Typical graphical representation of a schema

appropriate type. For example, the News programme will have a textual name and channel, and a date and time.

The type of a Collection (discussed later) is indicated by the end points of the curved line intersecting its *member relationship types* (the relationship types that relate an object type to its member object types). This intersection performs the function of identifying the member relationships types and in turn the member object types. It is especially salient when Collections have heterogeneous members. The style of the curved line (whether it is dotted, dashed, *etc.*) refers to the relationship key at the bottom of the diagram and indicates the relationship type that applies between member object types and in turn, the relationships that will be created between member objects. For example, the News reel sequence collection is an Attribute-Ordered Collection whose members are ordered on their start_frames. This is shown in the diagram by the curved line under the collection. This line has two arrow heads and an attribute name indicating that the collection is Attribute-Ordered.

4.6.1b Graphical representation of relationship types

The relationship types in the diagram are represented by straight lines whose end points indicate the base relationship type they are based on. Table 9 illustrates the four possible base relationship types and their typically associated end points on the right hand end of the line. The end points of the lines always touch the destination object type, and so indicate the direction of the relationship type.

Base relationship type	Graphical representation of the relationship type
Membership	——▶
Sequential	——▷
Temporal	——┤
Schema specific	——>

Table 9: Typical graphical representation of base relationship type

A key is present at the bottom of each diagram to indicate which lines represent which relationship types. Complex diagrams may also change the style of the line (dotted, dashed *etc.*) to provide unique lines for each relationship type, but if these become indistinguishable from each other then the names of relationships are written on the line. Figure 23 illustrates how the cardinality list of relationship types written next to their end points. For example, the cardinality list m is written next to the end point of the Contains relationship type from Anchor person sequence to Anchor person shot. This is not included in the key as several relationship types share the same name (and line style), but differ only in their cardinality list. The constraint specification of a relationship type is written below its name in the key (no constraint specifications are illustrated in the figure).

4.6.1c Graphical representation of collection type

Figure 23 only exemplifies two kinds of Collection: Pre-Ordered and Attribute-Ordered; the curved lines representing all Collection types are illustrated in Table 10. The direction of the arrow of the Pre-Ordered curved line indicates the linear sequence that will be present in the instance *i.e.* member objects will be ordered from left to right in the instance. For example, News programme is a Pre-Ordered Collection whose first member object will be a Title shot, which will be followed by one or more objects of type News report, these will then be followed by one Weather shot. An Attribute-Ordered Collection is represented by a curved line with an arrow at each end to indicate that the member objects are ordered in some way. The text just above the Attribute-Ordered curved line indicates which attribute is used to determine the linear sequence of the Collection. In the figure and table, the Attribute-Ordered Collections will order their member objects according to their start_frame attribute.

Table 10 also illustrates the graphical representation of Temporal, Un-Ordered, User-Ordered, and circular Collections. A Temporal Collection is represented by a curved line

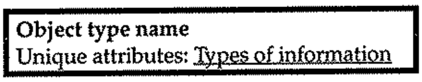
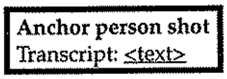
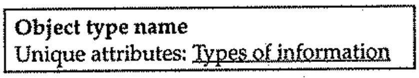
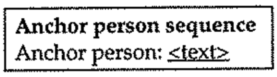
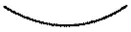
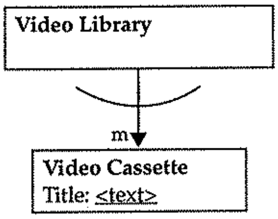
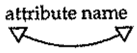
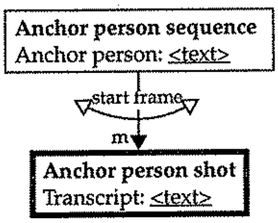
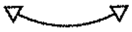
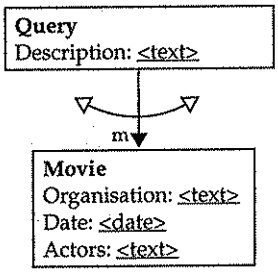
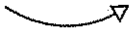
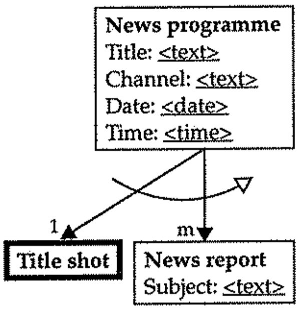
Base object type used	Graphical representation of the object type	Example
Unit		
Collection		
Un-Ordered Collection		
Attribute-Ordered Collection		 Anchor person sequence Anchor person: <text> start frame m Anchor person shot Transcript: <text>
User-Ordered Collection		 Query Description: <text> m Movie Organisation: <text> Date: <date> Actors: <text>
Pre-Ordered Collection		 News programme Title: <text> Channel: <text> Date: <date> Time: <time> 1 m Title shot News report Subject: <text>

Table 10: Graphical representations of object types

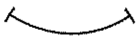
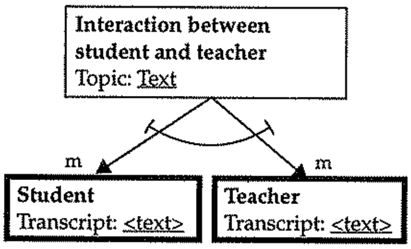
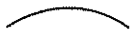
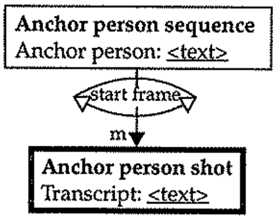
Base object type used	Graphical representation of the object type	Example
Temporal Collection		 Interaction between student and teacher Topic: Text m m Student Transcript: <text> Teacher Transcript: <text>
Circular Collection		 Anchor person sequence Anchor person: <text> start frame m Anchor person shot Transcript: <text>

Table 10: Graphical representations of object types

with 'T' ends intersecting the member relationship types. The table illustrates a representation of the usage of Temporal Collections discussed in Section 4.3.1. In this example the Temporal Collection is used to examine the temporal relationships between Student and Teacher interactions (here represented as Units). When the objects are instantiated, Temporal relationships will exist between the Student and Teacher objects which can then be examined. In this case the examination is used to draw out trends in the student and teacher interactions.

A User-Ordered Collection is represented by a curved line with two arrowed end-points. Unlike Attribute-Ordered Collections no attribute is named above the curved line. This is because the attribute that the User-Ordered Collection is ordered on (if any) is selected by the user when it is instantiated. These Collections are often used to store the result of queries; the example User-Ordered Collection in the table represents the example query discussed in Section 4.5. In the example the User-Ordered Collection will be ordered according to an attribute selected by the user, and will contain objects of type Movie that match the query 'movies produced by Universal in the 1960's in which Elvis did not appear'.

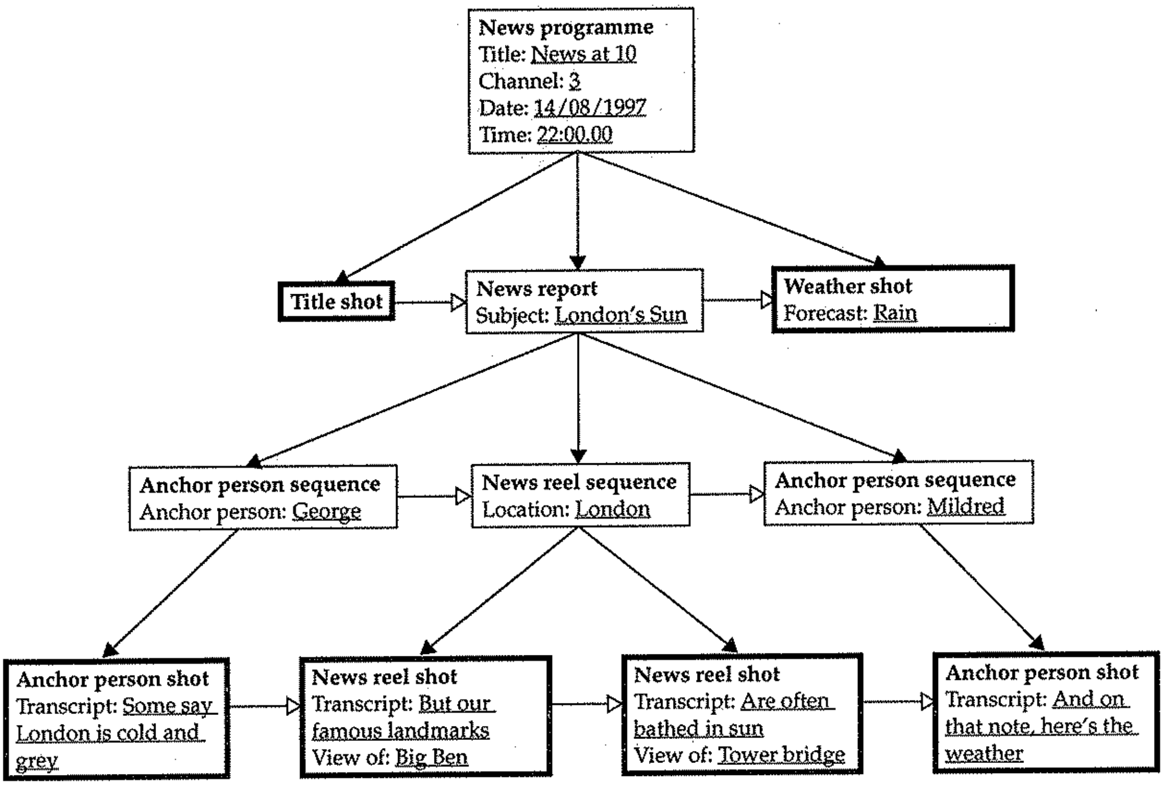
The table illustrates an Un-Ordered Collection which is represented by a curved line with no end points as there are no relationships between its member objects. The curved line is still present to be consistent with other Collections, and to saliently indicate which are the member relationship types. The example Un-Ordered Collection represents the Video Library discussed in Section 4.3.1. This Video Library contains an unordered collection of Video Cassettes.

Finally, an Ordered Collection which is circular is represented by adding a curved line above the curved line which represents the type of Collection. The table shows an example Attribute-Ordered Collection which is circular by virtue of the additional curved line. The presentation of the instance of this circular Collection would repeat until the user stopped it rather than stopping when the last member object had been displayed.

4.6.2 Graphical representation of instances

The graphical representation of instances is intentionally similar to the representation of schemas. Objects are represented by boxes and relationships are represented by straight arrowed lines. In keeping with Table 10, objects instantiated from Units are represented by thick edged boxes whereas objects instantiated from Collections are represented by thin edged boxes.

In an instance, the values of unique attributes are instantiated. Unique attribute values are shown in the representation as underlined text within object boxes. For example, in Figure 24, a News programme object has been instantiated with values for its unique attributes (Title: News at 10, Channel: 3, Date: 14/08/1997, Time: 22:00.00).



Relationship types	a ——▶ b	a ——▷ b
Name	*a* Contains *b*	*a* Precedes *b*
Inverse name	*b* is_within *a*	*b* follows *a*

Figure 24: Typical graphical representation of an instance

Unlike the schema representation, the instance representation does not need to describe the cardinality of links or the types of Collections. But it does show the relationships between objects using lines which use the same key as the schema diagram to maintain consistency between the schema and instance diagram. To recap, the end point of the line indicates the base relationship type, and different shading of lines may be used to distinguish them until they become indistinguishable in which case the name of the relationship appears on the line.

Figure 24 shows how the Contains and Precedes relationships of the Collections have been instantiated. For example, the News programme Pre-Ordered Collection is related to its member objects by the Contains relationship. Similarly, the member objects have Precedes relationships set up between them by VCMF to reflect the linear sequence of the Collection. In such a simple example the explicit representation of these relationships clutters the diagram, but in more complex instances discussed in Chapter 6 it is essential to explicitly represent the sequence.

4.7 Summary

This chapter has defined VCMF and its graphical representation. VCMF is a schema based framework (using a semantic data modelling approach) in which schemas define the characteristic structure and semantics of instances. Instances represent video content models using objects related to each other by relationships. Unlike object-oriented approaches, relationships are as important as objects, and are similarly instantiated from relationship types in the schema. Relationship types and object types are themselves instantiated from base types which VCMF uses to provide simple semantics. Objects in this semantic data modelling approach do no processing; it is the responsibility of VCMF support software such as the toolset discussed in Chapter 5 to maintain objects and relationships.

The reuse of schemas and instances has been touched upon in their definition; Chapter 6 uses the framework and graphical representation defined in this chapter to illustrate the use and reuse of schemas and instances.

Chapter 5 Supporting VCMF's Use

This chapter describes a toolset whose purpose is to support the creation and modification of instances and schemas, and the interactive use of these instances with their associated video. It is used in Chapter 6 to create, modify, and interact with, instances and schemas which are used to support the thesis defined in Chapter 3. It is also used to generate the diagrams of instances in Appendix D and schemas in Appendix E. Section 5.2 gives an overview of the toolset. The following sections describe the toolset's various component parts and their usage. They also present screenshots of the components in use.

5.1 Purpose of a Toolset to Support VCMF's Use

Video content models often contain thousands of relationships, segments, and annotations. For example, Taniguchi *et al.* [190] identified 168 segments in 12 minutes of video. A realistic example of video usage may use several hours of video recordings which, at this rate (around 850 segments per hour), may involve thousands of segments, let alone annotations and relationships.

Demonstrating the utility of this thesis (defined in Chapter 3) involves the use of several exemplar instances described in Chapter 6. Due to the complexity of these instances, it would be time consuming, error prone, and laborious to create and use them without an implemented toolset. Moreover, investigating reuse of video and its associated annotations would prove difficult, if not impossible, without an implemented toolset to support interactive use of instances. All exemplar schemas and instances discussed in Chapter 6 were created and manipulated using the toolset presented in this chapter.

This toolset's purpose is to support the demonstration of the validity of the thesis. In particular, supporting the demonstration of the use of VCMF to describe models and schemes, and to support reuse. To this end the toolset supports the following (there are certain aspects of VCMF not currently supported by the toolset, these are discussed in Chapter 7):

1. **Creation and manipulation of schemas** which are VCMF representations of video content modelling schemes. The basic features of schemas (object and relationship types) can be defined and manipulated.
2. **Creation and manipulation of instances** from these schemas; instances are VCMF's way of describing video content models. In the implementation objects and relationships between them can be created from schemas. Furthermore, certain collection based relationships can be automatically generated by the toolset.
3. **Interaction with instances** and their associated video. The toolset supports full interaction with instances, and relates objects to their associated video segments. This

interaction demonstrates the ability of VCMF to support real world video content modelling applications. Further, it supports the investigation of video and object reuse.

5.2 Toolset Overview

The toolset is *an* implementation to support the use of VCMF. Many other possible implementation approaches exist. This approach is inspired by the VideoSTAR platform [92] in which:

> "All tools in the system are based on a generic database platform called VideoSTAR (Video STorage And Retrieval) and they share video and meta data via the common database. The tools are managed by a video archive tool manager which provides mechanisms for communication and cooperation between different tools." [92]

VideoSTAR itself is not used as the basis of the VCMF toolset as it has a fixed video content modelling scheme which is a simple hierarchic structure (see Chapter 2). Following VideoSTAR's implementation approach increases the amount of implementation effort needed to create a simple video content modelling system; several different tools need to be constructed and communication between them needs to be supported. However, this approach is beneficial as it is flexible and allows additional interface tools to be easily added to the toolset. Such additional tools are typically tailored to different tasks which is important when considering using the toolset for further research activities (discussed in Chapter 8). Even demonstrating VCMF's abilities has involved the creation of a special interface for the video-map example in Chapter 6.

Figure 25 is an overview of the software components of the toolset (represented by boxes). These components communicate with each other using unidirectional sockets represented in the diagram by lines whose arrows indicate the direction of information flow. For completeness socket numbers are included in the diagram next to a curve which indicates the lines of communication using them. The grey boxes indicate the three main groups of components: Video User Interface components, Storage & Application Interface components, and User Interface components.

Three different communication protocols define the ways in which messages are structured and sent in the VCMF toolset. Two of the protocols are text based and use sockets. This provides a simple means of communicating between the components which are implemented using different platforms (C, Java, and SGI specific code). Moreover, it supports the possibility of using components written using other implementation platforms in the future. The protocols are represented in the diagram by the different line styles and are used for the following:

5.2.1 Database Protocol

Defined by Postgres [199] and used to transfer PSQL queries and results between the Instance & Schema Storage component and the Application Interface component.

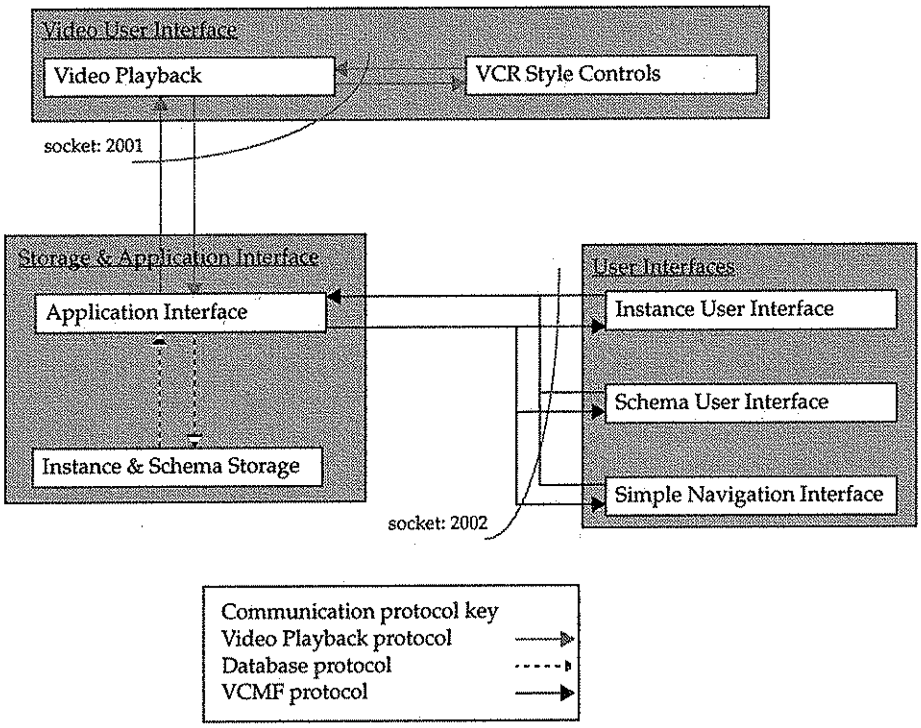


Figure 25: Toolset architecture

5.2.2 Video Playback Protocol

This protocol uses textual messages to control video playback, and to describe the playback state. It links the Application Interface, Video Playback, and VCR Style Controls components, and is fully described in Table 22, Table 23, and Table 24 of Appendix A. This protocol is much simpler than the VCMF protocol, and acts asynchronously without any replies to messages. Typical paraphrased messages are: *playback has stopped at frame 23, start playback at frame 45 and end at frame 78,* and *play video in reverse.*

5.2.3 VCMF Protocol

This protocol uses textual messages to communicate information about instances and schemas as well as the current state of video playback of objects. It is used between the Application Interface and User Interface components such as the Simple Navigation component. A brief overview of the VCMF protocol (fully described in Table 25 and Table 26 of Appendix A) follows. The protocol's purpose is to support communication about instances and schemas between components of the toolset. The protocol works in two communication modes: *synchronously* and *asynchronously.*

In synchronous communication a component sends a message to the Application Interface which then replies with a message. These messages are usually concerned with instances or schemas. For example, the Instance User Interface component may send a message requesting the creation of an object from a particular object type. The Application

Interface creates this object, finds a free ID for it, and stores information about it in the Instance & Schema Storage component. It then sends a reply message to the component informing it of the success of the operation. The information returned by the Application Interface is determined by a *mask* which must be present in the message from the component. This mask defines which attributes' values should be returned *e.g.* the value of the ID attribute and the first unique attribute of an object.

Asynchronous communication is used when the Application Interface needs to inform other components about events connected with the Video Playback component. The Application Interface broadcasts messages such as *playback has stopped at the end of object 47* to all components connected to it *via* sockets. Unlike synchronous communication, the components do not reply with a message.

The following section describes the Storage & Application Interface components which do not directly interact with the user. This is followed by a description of the Video User Interface components which present video to the user and allow simple (linear) control over the video playback. Finally, the User Interface components are described. These allow the user to manipulate instances and schemas, and to navigate through instances and their associated video.

5.3 Storage & Application Interface Components

This section describes two components: the Instance & Schema Storage component, and the Application Interface component. These components are the central repository and communication components of the toolset. As a result they are essential to the toolset's use.

5.3.1 Instance & Schema Storage

This component stores VCMF schemas and instances. These are stored in four tables of a Postgres database [199] whose names, purpose, and field names are outlined in Table 11. Appendix B fully details these tables, field names, and uses; Appendix C presents the content of the database containing all instances and schemas used in this thesis.

Table name	Information stored	Field names
schema_objects	object types	SID, Name, Kind, Author, Temporal, Indexed_children, Ordered_children, App_info, App_fields
schema_links	relationship types	SID, Name, Reverse_name, Navigation, Source_def, Dest_def, Author
instance_objects	objects	SID, ID, Name, Start_frame, End_frame, Children, Representative, Creator, App_info
instance_links	relationships	SID, ID, Source, Destination, Creator, GroupID

Table 11: Postgres tables used to store schemas and instances

With the exception of unique attributes, all attributes are stored in their own fields in tables. For example, the SID field of the schema_objects table stores the unique identifier of the object type (referred to as the TID in Chapter 4).

Unique attributes are handled differently because it is cumbersome to add fields to tables once they have been created. Basically the schema_objects table holds an array of unique attribute names in its App_Info field. The unique attribute value types are held in its App_fields field. The values of unique attributes of objects are stored in an array in the App_Info field of the instance_objects table. For simplicity, the toolset automatically gives each object a key unique attribute called Name with value type <text>.

In the current implementation of the toolset the unique attribute values are simply textual descriptions. No checking of their validity is performed by the toolset. This compromise is briefly returned to in Chapter 7.

The Representative field of the instance_objects table stores representative information about objects. In the current implementation this representative information is always a frame number. The frame identified by this frame number is assumed to be representative of the whole object. This is not necessarily the best way of representing the content of an object, but many of the systems outlined in Chapter 2 use this approach so it was felt appropriate for this prototype implementation.

Postgres is a relational database which has no concept of *relationships* in the sense used in this thesis *i.e.* the relationship that exists between two objects. However, it is used to store data about schemas and instances as it is readily available, robust, and provides an application interface which is simple to use. The next subsection describes the construction of the Application Interface component which interacts between other software components and this database.

5.3.2 Application Interface

This component provides the communication hub between components. It provides the interface between the Instance & Schema Storage component, the Video Playback component, and User Interface components. In addition, this component deals with the relationship maintenance, and query support tasks involved in using VCMF instances. The VCMF communication protocol is used to support communication between components about instances and schemas.

The Application Interface component communicates with the Postgres database described in the previous subsection using the Postgres library libpq.h provided with the database. The Application Interface component's main task is to translate requests from other components into PSQL [199] queries, and in turn translate the results of such queries into the VCMF protocol. Other components do not directly communicate with the database as its does not directly represent instances and schemas; the Application Interface interprets the content of the database and translates them into representations of instances and schemas. As such it effectively acts as a translator between the database protocol and the VCMF protocols.

5.3.2a Relationship maintenance tasks

As mentioned previously, this component deals with instance maintenance tasks. These are necessary as VCMF provides a semantic data model and so objects have no processing associated with them (unlike object-oriented approaches).

The Application Interface is responsible for instantiating all necessary relationships between member objects of a Collection. This relates to Garzotto *et al.*'s [71] third advantage of schema based approaches discussed in Chapter 4. This automatic instantiation of relationships is achieved by first inspecting the object type of the Collection and determining whether any relationships should be instantiated. VCMF states that relationships between members can only be automatically instantiated if the Collection is Attribute-Ordered or Temporal; the current toolset only supports automatic relationship instantiation for Attribute-Ordered Collections. If relationships can be automatically instantiated then the object type determines what, if any, relationship types should be instantiated. The member objects are then inspected to determine which relationships should be instantiated between which objects.

For example, Chapter 6 illustrates the use of queries to create new collections of objects. When the query is completed and the resultant objects have been added to the collection, relationships may need to be instantiated between these member objects. In this example Precedes relationships are instantiated between member objects based on their Approximate Date attribute. Figure 26 shows the Query Collection on the left and some of its members objects on the right. The Contains relationships (defined in the schema illustrated in Chapter 6) are instantiated between the Query Collection and the member objects by this component. The Precedes relationships (defined by the Query object type illustrated in Chapter 6) are then instantiated between member objects as appropriate. This is implemented as follows:

- The Application Interface component queries the Instance & Schema Storage component about the appropriate ordering of the member objects using psql.
- The Instance & Schema Storage component returns a list of the objects ordered appropriately (in this example by their Approximate Date attribute).
- The Application Interface component then iterates through this list and instantiates the Precedes relationships between member objects (shown in Figure 26).

5.3.2b Query formulation

The previous example also illustrates this component's support for query formulation using SVQL. As mentioned above, this component not only poses the query to the Instance & Schema Storage component using psql, but also creates the appropriate Collection and relationships between objects. Again, this shows how the Application Interface provides the bulk of the interpretation of VCMF in the toolset. It is

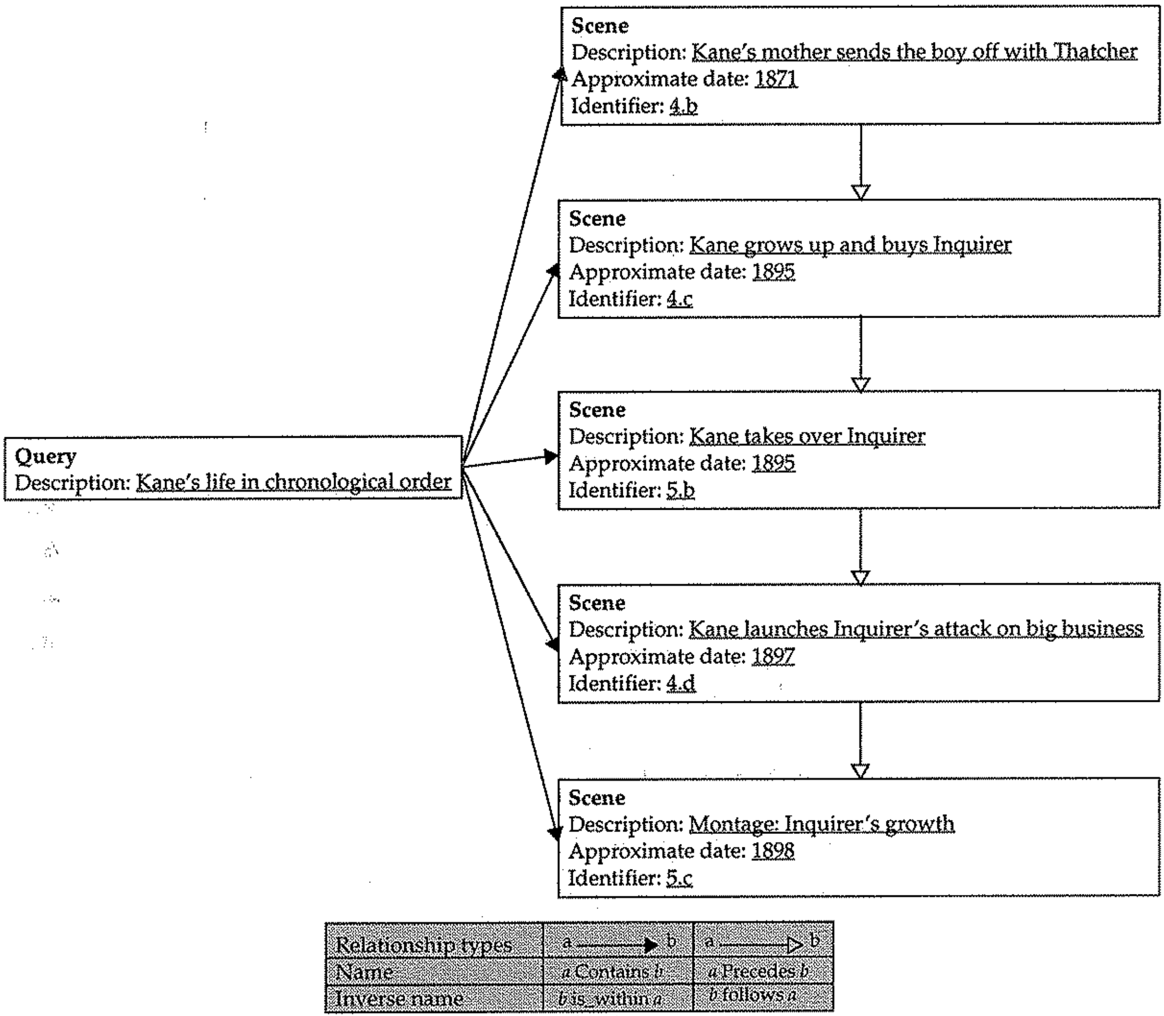


Figure 26: Extract of result of the query discussed in Chapter 6

responsible for creating and maintaining the objects and their relationships, as well as supporting higher levels functions such as querying instances.

5.3.2c Integrity checks

Another maintenance task supported by this component is an integrity check of instances. These checks relate to the first advantage of schema based approaches discussed by Garzotto *et al.* [71] and presented in Chapter 4. The component performs the following checks on the instances with respect to their schemas:

1. Are all the object attribute values filled in appropriately? *i.e.* do they satisfy their value type specification?
2. Are relationships instantiated between appropriate objects?
3. Do relationships have the appropriate cardinality?
4. Are relationships' constraints lists satisfied?

If any of these questions lead to a negative result the Application Interface generates a message. This is accessed from other components by checking the list of pending messages. These messages detail the questions that could not be satisfied, the reason why they could not be satisfied, and the objects and/or relationships involved. Currently this integrity check takes a very long time and so is rarely used though future implementations may increase the efficiency of the checking algorithms. Improved checking algorithms could support real time checking of instances during construction rather than *ad-hoc* checking of larger instances.

5.3.2d Communicating video based events

This component is also responsible for communicating video based events such as *playback stopped* to other components. It translates Video Playback protocol messages into VCMF protocol messages as necessary. For example, when playback stops, the Video Playback component informs the Application Interface using the Video Playback protocol. The Application Interface then informs User Interface components that playback has stopped using the VCMF protocol (including details of which object is affected). Navigation interfaces are typically interested in such messages so that they can inform the user of the current state of playback with respect to the instance, and inform them of choices they may make to influence the playback of video.

5.3.2e Controlling video playback

Finally, this component is also responsible for controlling the video playback with respect to objects and relationships of instances. At the simplest level, it handles the playback of video segments associated with objects by informing the Video Playback component of the frames to be played. It also handles the sequencing of segments to be played by following relationships between objects. For instance, objects related to each other by Precedes relationships are sequenced in the following steps:

1. The Application Interface component sets the current object to be the first object in the sequence.
2. The Application Interface component informs the Video Playback component of the segment to be presented (the segment of the current object).
3. When the segment has been presented the Video Playback component informs the Application Interface component.
4. The Application Interface component locates the next object in the sequence by querying the Instance & Schema Storage component for the object which is the destination of a Precedes relationship which has the current object as its source.
5. If the Instance & Schema Storage component locates a matching object the Application Interface components sets the current object to be it, and the process returns to step 2. Otherwise the end of the sequence has been reached.

5.4 Video User Interface Components

These components display video, and provide the user with simple VCR style controls to control the video's linear presentation. They do not manipulate instances or schemas in any way. However, as discussed previously, the Video Playback component may be controlled by the Application Interface component.

5.4.1 Video Playback

The Video Playback component displays video in a window. A screenshot of a typical image displayed to the user is given in Figure 35b which shows the small size and low quality of the video presented. The quality of the video image is sacrificed to allow large amounts to be stored. Currently approximately 147 minutes are stored in about 288 Mb.

This component is only controlled by other components and accepts no input from users. User Interface components influence the playback of video, and are informed of the playback's state (*e.g.* when the video comes to the end of a segment) *via* the Application Interface component. This component itself communicates with the Application Interface and the VCR Style Controls components using the Video Playback protocol discussed previously.

All video is stored in one large file which is selected when the component is started. This file is currently approximately 288.2 Mb in size, contains 70918 frames, and is stored on a disc directly connected to the computer that the component is executed on in order to minimise data transfer delays. Frame rates of up to 10 frames per second are achievable with this arrangement. Storing video on a network accessed disc involves network transmission delays which reduce the frame rate to approximately 2 frames per second. This is typically unsuitable for useful video playback [47].

5.4.2 VCR Style Controls

This component provides the user with simple VCR style controls such as *play, fast forward, stop etc.* illustrated in Figure 27. This screenshot shows the various playback controls in the middle row of buttons, the information about the current playback frame at the top (here frame 1255), and various playback options at the bottom including the ability to move the video to specific frames (here frame 1234). The component is typically used to define video segments when creating and editing objects in instances. When navigating through completed instances, a navigation interface is usually used instead of the simple VCR controls.

This component only communicates with the Video Playback component and uses the Video Playback protocol. If fact, the Video Playback and VCR Style Control components could be executed together without the rest of the VCMF toolset, but this provides none of the benefits of the instances and schemas.

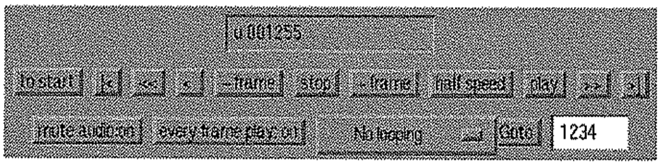


Figure 27: VCR style controls

5.5 User Interface Components

The User Interface components present information about instances and schemas to the user and allow them to be manipulated. Furthermore, they support navigation within instances and their associated video. This shows the viability of the implementation approach taken in this thesis.

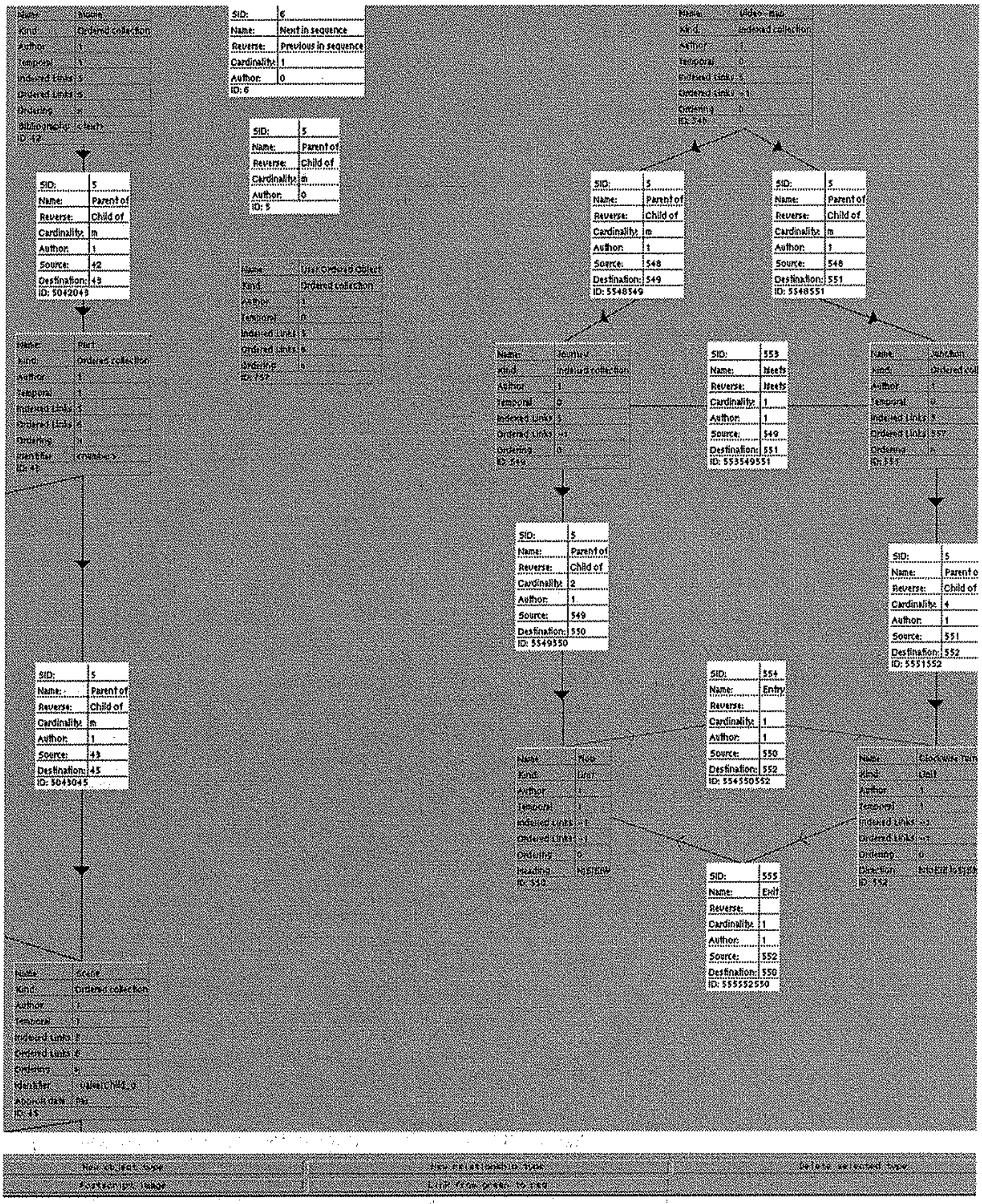
The set of User Interface components illustrated in Figure 25 is the minimum configuration needed to create and use schemas, instances, and associated video. Naturally, some tasks such as navigating around an instance may not need all components to be co-present; there is no compunction to always have all available components running at the same time. Furthermore, additional interface components can be added and removed as necessary; an example of this is presented in Section 5.5.4.

All User Interface Components are written in Java [81] as it supports relatively quick and easy user interface construction. The drawback of using Java is the poor run time performance, but the toolset is intended to support the demonstration and investigation of the use of VCMF, not to be a commercially viable product.

5.5.1 Schema User Interface

This provides the user with a means of creating and manipulating schemas. Figure 28 shows a typical screenshot of the component. The main area of the window displays a representation of schemas. All available schemas are displayed at the same time in this area which can be scrolled as necessary. This example shows the toolset's representation of the plot segmentation and video-map schemas. The grey boxes represent object types, and the white boxes represent relationship types whose source and destination types are indicated by the lines on the diagram. White boxes with no connecting lines represent base relationship types. Lines are arrowed to indicate the direction of the relationship; the arrow style indicates the base relationship type used. The text in the boxes presents the attribute names on the left and value types on the right.

For example, in the plot segmentation schema of Chapter 6 there is an object type named Film which is an Attribute-Ordered Collection (ordered on the start_frame) which has a unique attribute called Bibliography, and a key unique attribute called Name. The toolset's representation of this object type is the grey box in the top left hand corner of the figure. The text in the box details the object type's name (Movie in this case), its base type



Name: Movie
Kind: Ordered collection
Author: 1
Temporal: 1
Indexed Links: 5
Ordered Links: 5
Ordering: x
Bibliography: <text>
ID: 42

SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 42
Destination: 43
ID: 5042043

Name: Part
Kind: Ordered collection
Author: 1
Temporal: 1
Indexed Links: 5
Ordered Links: 6
Ordering: x
Identifier: <number>
ID: 43

SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 43
Destination: 45
ID: 5043045

Name: Scene
Kind: Ordered collection
Author: 1
Temporal: 1
Indexed Links: 5
Ordered Links: 6
Ordering: x
Identifier: value(child_o
Approx date: Pts
ID: 45

SID: 6
Name: Next in sequence
Reverse: Previous in sequence
Cardinality: 1
Author: 0
ID: 6

SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 0
ID: 5

Name: User Ordered Object
Kind: Ordered collection
Author: 1
Temporal: 0
Indexed Links: 5
Ordered Links: 6
Ordering: n
ID: 157

Name: Video-map
Kind: Indexed collection
Author: 1
Temporal: 0
Indexed Links: 5
Ordered Links: -1
Ordering: 0
ID: 548

SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 548
Destination: 549
ID: 5548549

SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 548
Destination: 551
ID: 5548551

Name: Journey
Kind: Indexed collection
Author: 1
Temporal: 0
Indexed Links: 5
Ordered Links: -1
Ordering: 0
ID: 549

SID: 553
Name: Meets
Reverse: Meets
Cardinality: 1
Author: 1
Source: 549
Destination: 551
ID: 553549551

Name: Junction
Kind: Ordered coll
Author: 1
Temporal: 0
Indexed Links: 5
Ordered Links: 557
Ordering: n
ID: 551

SID: 5
Name: Parent of
Reverse: Child of
Cardinality: 2
Author: 1
Source: 549
Destination: 550
ID: 5549550

SID: 5
Name: Parent o
Reverse: Child of
Cardinality: 4
Author: 1
Source: 551
Destination: 552
ID: 5551552

SID: 554
Name: Entry
Reverse:
Cardinality: 1
Author: 1
Source: 550
Destination: 552
ID: 554550552

Name: Run
Kind: Unit
Author: 1
Temporal: 1
Indexed Links: -1
Ordered Links: -1
Ordering: 0
Heading: NSEW
ID: 550

Name: Clockwise Turn
Kind: Unit
Author: 1
Temporal: 1
Indexed Links: -1
Ordered Links: -1
Ordering: 0
Direction: NSEWNSEW
ID: 552

SID: 555
Name: Exit
Reverse:
Cardinality: 1
Author: 1
Source: 552
Destination: 550
ID: 555552550

Figure 28: Extract of a typical screenshot of the Schema User Interface component

(Ordered Collection), the ordering of the collection (x indicates start_frame so it is Attribute-Ordered on the start_frame, see Appendix B for full details of the toolset's representation of ordering attributes), a number representing the person who created it (1), and the unique attribute Bibliography with its type <text>. Note that, for simplicity, in the toolset each object type defines the key unique attribute to be called Name and have a value type <text>; this is not shown on the diagram.

The object type discussed in the previous paragraph is the source object type of the relationship type just below it. This relationship type's attributes are presented as text in the white box, and its destination is the object type called Part. It is based on the base relationship type Parent_of which is indicated by its Name, its SID (referred to as TID in Chapter 4), and the style of the arrows on its connecting lines (here filled black arrows). Its other attributes are: its unique identifier (ID) the SIDs of the source and destination types (in this example 42 and 43 respectively), a number representing the person who created it (here 1), and details of the cardinality of the relationship (here m to indicate 1-m cardinality).

The base relationship type Parent_of is represented by the white, unconnected box at the top middle of the screenshot, just below the white box representing the base relationship type Next_in_sequence. This defines the name, inverse name, default cardinality, and line style of relationship types created from it.

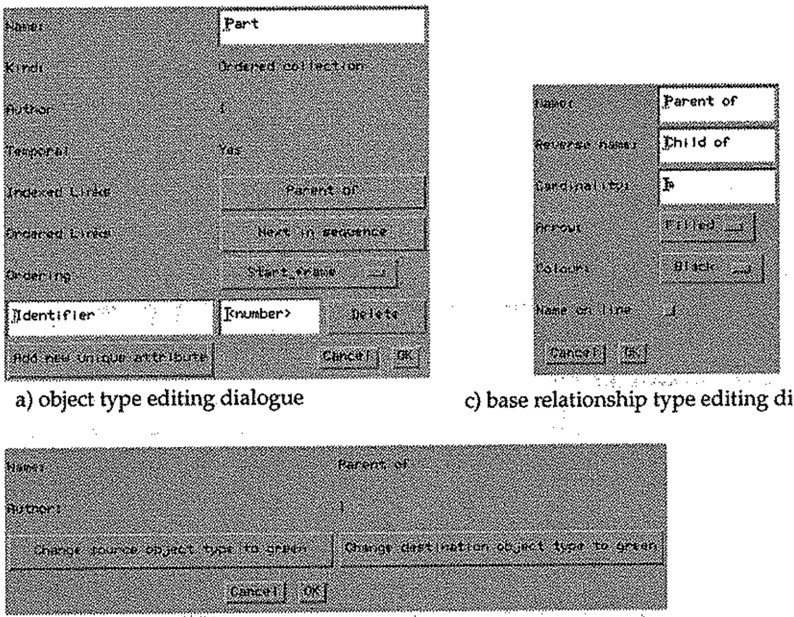
5.5.1a User interaction

Users interact with the schema either by using the buttons at the bottom of the screen, or by clicking on the boxes themselves. Clicking on a box selects it and causes it to be highlighted by a surrounding red box. The previously selected box (if any) is surrounded by a green box which allows operations to be performed between two boxes (such as setting up a relationship type between two object types). All other boxes have no surrounding coloured box. Clicking and keeping the mouse button down (dragging) causes the box to follow the mouse pointer so long as the button is held down. Double clicking on a box pops up a dialogue box whose content depends on whether the box represents an object or relationship type.

Double clicking on an object type pops up a dialogue box which allows the user to add and remove unique attributes. If it is a Collection it also allows the user to select the relationship type that will exist between member objects. Figure 29a is a screenshot of such a dialogue. The user can:

- Edit the name of the object type (the key unique attribute, here it is Part).
- Change the type of the relationships that will exist between member objects (here Next_in_sequence). This can only be changed if the object type is a Collection.
- Change the ordering of the members using the list of attributes (start_frame is currently selected). This is only applicable if the object type is an Attribute-Ordered Collection.
- Add, amend, and delete unique attributes (here the only one is called Identifier and has value type <number>).

Figure 29b shows the dialogue presented when relationship types are double clicked. It allows the user to change the source or destination object types to the previously selected object type (highlighted in green).



Name: Part
Kind: Ordered collection
Author: i
Temporal: Yes
Indexed Links: Parent of
Ordered Links: Next in sequence
Ordering: Start_frame
Identifier <number>

Delete
Add new unique attribute
Cancel
OK

a) object type editing dialogue

Name: Parent of
Reverse name: Child of
Cardinality: m
Arrow: Filled
Colour: Black
Name on line

c) base relationship type editing dialogue

Name: Parent of
Author: i

Change source object type to green
Change destination object type to green
Cancel
OK

b) relationship type editing dialogue

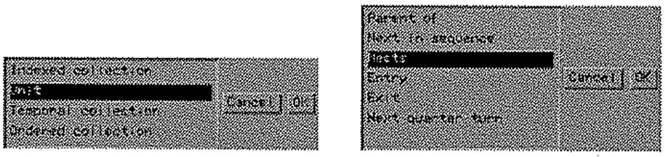
Figure 29: Dialogue boxes popped up when types are double clicked

Finally, double clicking on a base relationship type (a white box with no connecting lines) pops up the dialogue box in Figure 29c. This allows the user to:

- Change the name and inverse name of the base relationship type (here Parent_of and Child_of respectively).
- Define the cardinality of the relationship type (here m indicating a 1-m cardinality).
- Define the way in which the connecting lines are drawn in the Instance and Schema User Interface components. In this example the base relationship type is represented by a black, filled arrow, and its name is not drawn next to the line.

The buttons at the bottom of the screen provide the following functionality:

- **New object type**: this supports the creation of new object types. It pops up a dialogue box which allows the user to select the base object type of the new object type. Figure 30a is a screenshot of this dialogue; in this diagram the user has selected the base object type Unit. Once this is selected the object type is created and the user is presented the object type editing dialogue discussed previously and illustrated in Figure 29a.
- **Link from green to red**: this is creates a relationship type from one object type to another. The source object type is the previously selected object type (highlighted in green), and the destination object type is the currently selected object type (highlighted in red). When the user clicks on this button they are presented with a dialogue in which they select the base relationship type that the new relationship type will be based on. Figure 30b is a screenshot of this dialogue in which the user has



a) select base object type b) select base relationship type

Figure 30: Dialogues to select base object and relationship types

selected the Meets base relationship type. After selecting this the relationship type is created between the two object types.

- **New base relationship type**: this supports the definition of base relationship types. When clicked this button creates a blank base relationship type and pops up the base relationship editing dialogue discussed previously and shown in Figure 29c.
- **Delete selected type**: when clicked, the currently selected object or relationship type is deleted from its schema.
- **Postscript image**: this produces a postscript [4] representation of all schemas. When clicked, it pops up a dialogue prompting the user to select a file to be the receptacle of a postscript representation of all schemas. The diagrams in Appendix D and Appendix E were all generated in this way. Such diagrams are much clearer than screenshots which quickly become difficult to read due to pixellation, and large diagrams require the joining of many screenshots.

5.5.1b Implementation approach

This component is implemented in Java; each object type and relationship type box is a Java object which renders its text and box to the screen or a postscript file as appropriate. Java objects representing relationship types additionally render the lines between themselves and their source and destination object types. These Java objects are held in a list which is used to request all objects to render themselves on the appropriate medium, and to inform objects of mouse clicks. When a mouse click happens within a Java object the object pops up the appropriate dialogue box or changes its screen position. The highlighting of the objects is handled separately by the main drawing object. The Java objects are representations of the data held in the Instance & Schema Storage component; as the user creates and manipulates the Java objects this information is relayed to the Instance & Schema Storage component *via* the Application Interface. Although this means that there are two representations of the same information, the data stored in the Instance & Schema Storage component is always regarded as being *bona fide*.

5.5.2 Instance User Interface

This component supports the creation and manipulation of instances. It shares much of its implementation with the Schema User Interface component, and displays information in a similar manner. Figure 31 shows an extract of a screenshot displaying part of the instance of plot segmentation of *Citizen Kane* discussed in Chapter 6. Like the Schema User Interface there is a large display area in which boxes and lines represent instances (all instances are displayed). Below this are some buttons to allow manipulation of the instances. Grey boxes represent objects, and the lines represent relationships between objects (the small white boxes are *handles* for relationships which allow them to be edited). The grey boxes contain text representing attribute names on the left, and values on the right. The arrows on lines represent the direction of the relationship, and their style indicates their relationship type (defined in the Schema User Interface).

5.5.2a User interaction

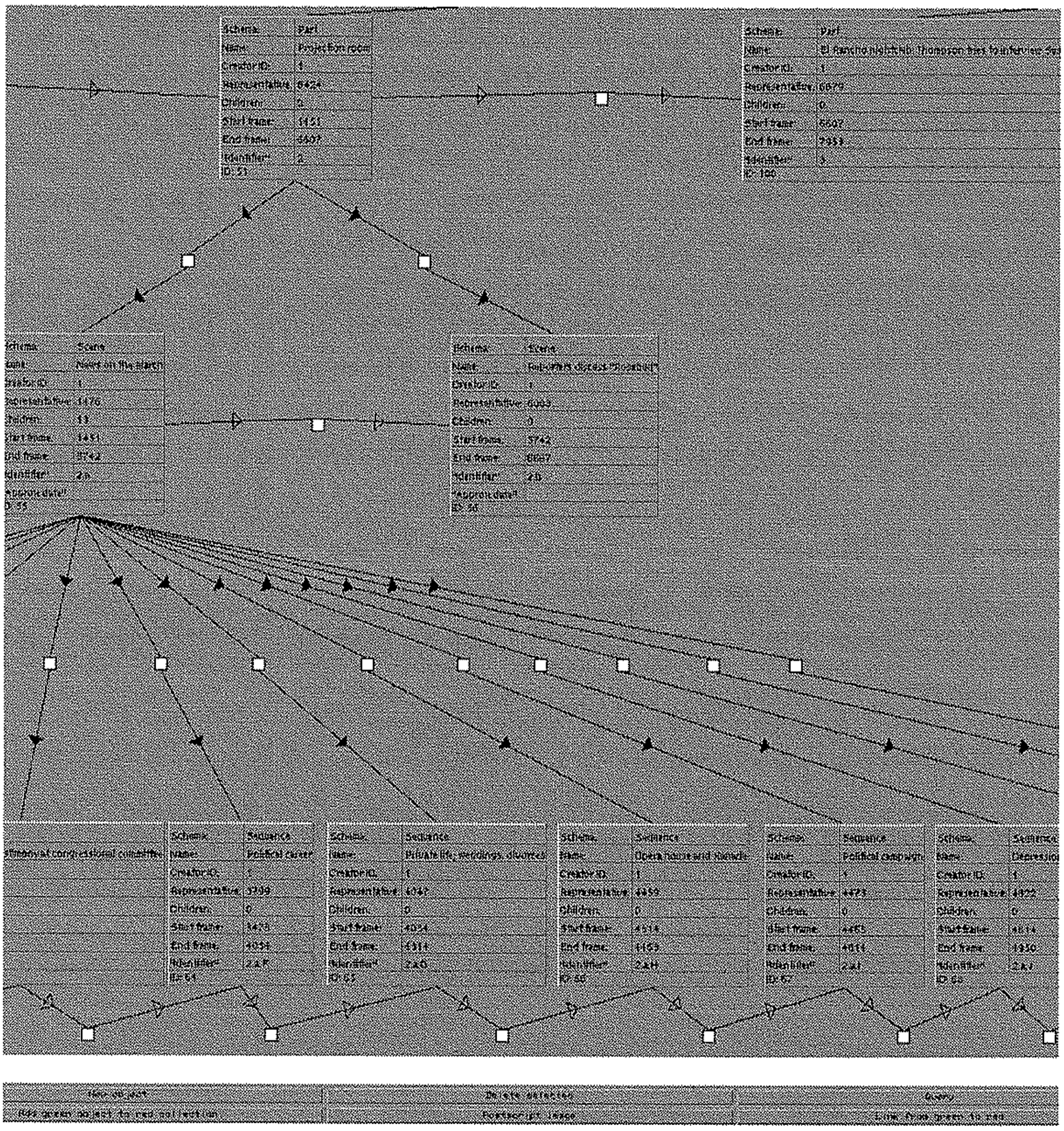
In keeping with the Schema User Interface, users interact with the instance either by clicking on boxes, or clicking buttons at the bottom of the screen. Double clicking on an object causes a dialogue to pop up allowing the user to edit values of the object's attributes. Figure 32a shows the dialogue box displayed to allow users to edit an object's attributes. Here the object is of type Part, has the Name (the key unique attribute) Projection Room, and the unique attribute Identifier has value 2. The dialogue allows the user to set the start, end, and representative frames of the object by clicking the appropriate button. The frame number used is the number of the frame currently being displayed by the Video Playback component.

Double clicking on a relationship box (white) pops up a dialogue box to allow the user to change the source or the destination object of the relationship to the previously selected object. Figure 32b shows the dialogue box popped up when a relationship box is double clicked. Here the relationship Parent_of can have its source and destination object changed to the previously selected object (outlined in green).

As in the Schema User Interface component, clicking once on an object selects it (indicated by a red border), and causes the previously selected object to have a green border; all other objects have no coloured border. Clicking and dragging an object allows the user to reposition it.

The following buttons are present at the bottom of the screen:

- **New object**: supports the creation of a new object. When it is clicked, a dialogue is presented from which the user chooses the object type that the new object should be created from. Figure 33a shows this dialogue. When the object type has been selected, the object is created and the user is presented with the object editing dialogue previously discussed and shown in Figure 32a. The attributes themselves are defined by the object type selected.



Schema:	Part
Name:	Projection room
Creator ID:	1
Representative:	6424
Children:	2
Start frame:	1451
End frame:	6607
Identifier*:	2
ID: 51	

Schema:	Part
Name:	El Rancho nightclub: Thompson tries to interview Su
Creator ID:	1
Representative:	6679
Children:	0
Start frame:	6607
End frame:	7953
Identifier*:	3
ID: 100	

Schema:	Scene
Name:	News on the march
Creator ID:	1
Representative:	1476
Children:	11
Start frame:	1451
End frame:	5742
Identifier*:	2.a
approx date	
ID: 55	

Schema:	Scene
Name:	Rawlston directs "Rosebud"
Creator ID:	1
Representative:	6093
Children:	0
Start frame:	5742
End frame:	6607
Identifier*:	2.b
approx date	
ID: 56	

	stimony at congressional committee

Schema:	Sequence
Name:	Political career
Creator ID:	1
Representative:	3799
Children:	0
Start frame:	3478
End frame:	4034
Identifier*:	2.a.F
ID: 64	

Schema:	Sequence
Name:	Private life; weddings, divorces
Creator ID:	1
Representative:	4047
Children:	0
Start frame:	4034
End frame:	4314
Identifier*:	2.a.G
ID: 65	

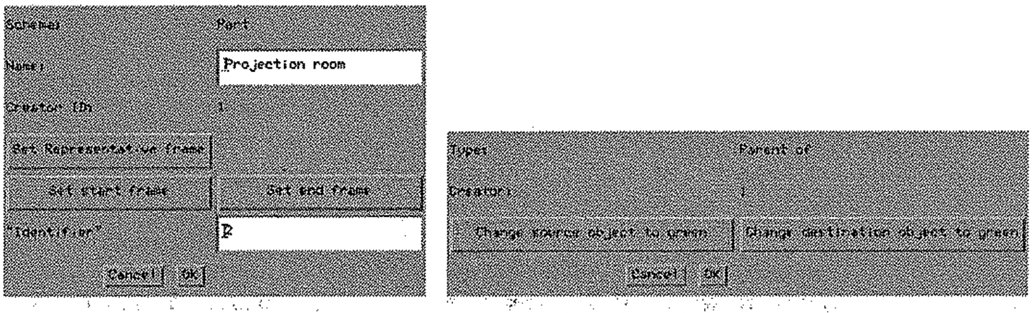
Schema:	Sequence
Name:	Opera house and Xanadu
Creator ID:	1
Representative:	4459
Children:	0
Start frame:	4314
End frame:	4453
Identifier*:	2.a.H
ID: 66	

Schema:	Sequence
Name:	Political campaign
Creator ID:	1
Representative:	4473
Children:	0
Start frame:	4453
End frame:	4614
Identifier*:	2.a.I
ID: 67	

Schema:	Sequence
Name:	Depressio
Creator ID:	1
Representative:	4822
Children:	0
Start frame:	4614
End frame:	4930
Identifier*:	2.a.J
ID: 68	

Figure 31: Extract from a screenshot of the Instance User Interface component

- **Link from green to red**: supports the creation of a relationship from one object to another. If two objects are currently selected, clicking this button causes a dialogue to appear with which the user selects the relationship type from which the new relationship is to be created. A relationship of the selected type is then instantiated between the previously selected object (the source), and the currently selected object (the destination). Figure 33b shows an example of this dialogue box detailing the relationship types available. As relationships involved with Collections are automatically instantiated and maintained by the Application Interface component, this button is typically used to instantiate Schema specific relationship types.



Schema: Part

Name: Projection room

Creator ID: 1

"Identifier"

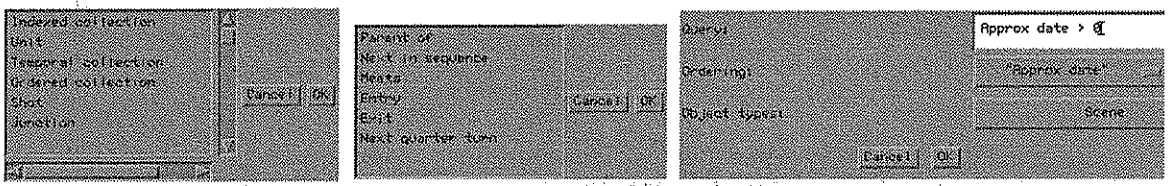
Type: Parent of

Creator: 1

a) object editing dialogue

b) relationship editing dialogue

Figure 32: Object and relationship editing dialogue



Query: Approx date > 0

Ordering: "Approx date"

Object types: Scene

a) object type selection

b) relationship type selection

c) query formulation dialogue

Figure 33: Dialogues to select object and relationship types, and to construct queries

- **Delete selected**: deletes objects or relationships. When pressed it deletes the currently selected object or relationship. The Application Interface component ensures that any relationships left without a source or destination object after such an operation are deleted as well.
- **Query model**: facilitates querying of instances. This button is the main difference between this component and the Schema User Interface component. When pressed, a dialogue (shown in Figure 33a) is presented which allows the user to select the object type of the Collection that will be created to hold the result of the query (the into part of a SVQL query; see Chapter 4 for details of SVQL). Once the object type is selected, the user formulates their query using the dialogue illustrated in Figure 33c. They specify the ordering, types of objects to be queried, and the query and then submit the query. Respectively, these specifications constitute the order by, from, and where parts of an SVQL query. On completion of the operation, a newly created Collection and its associated relationships appear in the display of the instances. In the example dialogue of Figure 33c the user has set up a query to create a Collection containing Scenes (selected in the Object Types list) from the *Citizen Kane* plot segmentation ordered by their Approximate date attribute (selected in the Ordering list). The query here is simply Approximate date > 0 which ensures that only objects which have a value for the Approximate date attribute are included in the Collection. Assuming that the user had initially selected a Query object type, the SVQL form of

this query would be: select from Scenes into Query where Approximate date > 0 order by Approximate date.

- **Add green object to red collection:** this adds the previously selected object to the currently selected Collection. The Application Interface component determines which relationships (if any) need to be instantiated between member objects and instantiates them.
- **Postscript image:** as with the Schema User Interface, this produces a postscript representation of the instances.

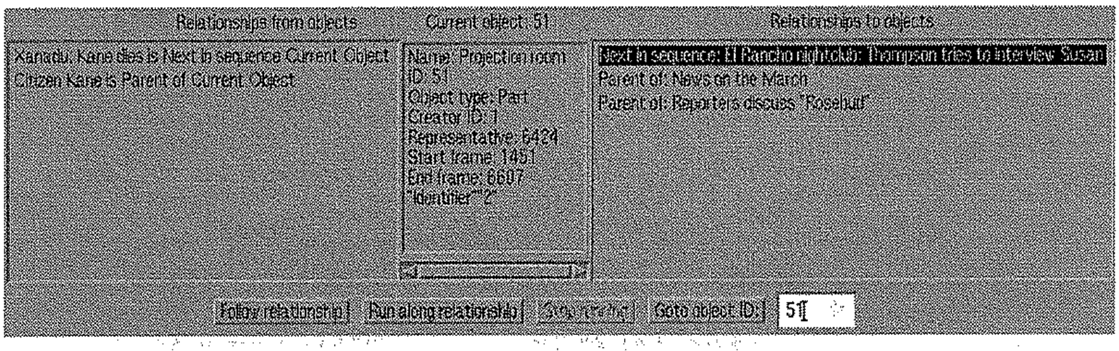
5.5.3 Simple Navigation Component

This component supports a user's navigation within instances *via* relationships of the instance. The user is presented with details of the *current* object's attributes as well as relationships leading to and from it. The user can then select a relationship to follow in one of two ways: either simply going to the object at the other end of the relationship and displaying its associated video, or *running* along relationships of that type.

Running along relationships means that the object at the end of the relationship is made the current object and its associated video is played. Once the associated video playback has finished (the playback reaches the object's end_frame), the system attempts to change the current object to be the one at the end of the relationship of the same type as the previous, in the same collection as the previous, and *heading* in the same direction as the previous. The heading of the relationship refers to whether the current object is the source (heading: *from*) or the destination (heading: *to*) of the relationship. Ensuring the same heading prevents the system simply going backwards and forwards along the same relationship. As discussed in Section 5.3, the Application Interface component handles this form of navigation; it is not the Simple Navigation component's responsibility.

Figure 34 is a screenshot of the Simple Navigation component. It shows three main information windows which are, from left to right: relationships that lead *to* the current object (the current object is their destination) with their source object's key unique attribute value, attributes of the current object, and relationships that lead *from* the current object (the current object is their source) with their destination object's key attribute value. Double clicking on one of the relationships causes the relationship to be followed.

In this example the current object is of type Part, it has ID 51, is called Projection room (the key unique attribute) and has one unique attribute named Identifier with value 2. The figure also details the Representative_frame, Start_frame, and End_frame attributes of the object. This object is lead to by the relationships: Parent_of (from the object called Citizen Kane) and Next_in_sequence (from Xanadu: Kane dies). Conversely, the object is the source object of the relationships Parent_of (leads to News on the March), Parent_of (leads to Reporters discuss "Rosebud"), and Next_in_sequence (leads to El Rancho Nightclub: Thompson tries to interview Susan).



Xanadu: Kane dies is Next in sequence Current Object
Citizen Kane is Parent of Current Object

Name: Projection room
ID: 51
Object type: Part
Creator ID: 1
Representative: 6424
Start frame: 1451
End frame: 6607
"Identifier""2"

Next in sequence: El Rancho nightclub: Thompson tries to interview Susan
Parent of: News on the March
Parent of: Reporters discuss "Rosebud"

Figure 34: Screenshot of the Simple Navigation component

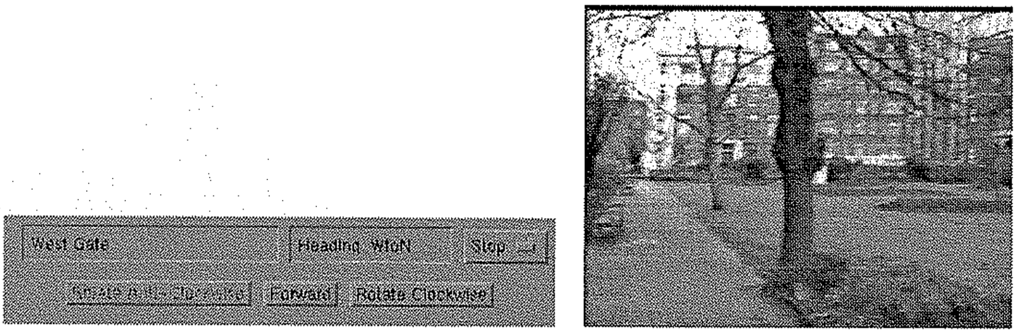
This object can be seen in the diagram in Appendix D (pp. 278), and in Figure 31. In Figure 31 the relationships to and from this object (ID 51), and its attributes are shown. The Parent_of relationships are shown by filled arrows on the lines, and the Next_in_sequence relationships are shown by the empty arrows.

The buttons at the bottom of the screenshot allow the user to:

- **Follow relationship**: the currently selected relationship is followed. In Figure 34 the relationship Next_in_sequence (leading to El Rancho Nightclub: Thompson tries to interview Susan) is currently selected. Clicking this button causes the current object to be changed to the object at the other end of the relationship, and its associated video played.
- **Run along relationship**: the currently selected relationship is *run* along as previously described.
- **Stop running**: this stops the running along of a relationship, and stops the associated video playback. In this figure this button is greyed out to indicate that it has no function in the current navigation state as no link is being run along.
- **Goto ID**: jumps to a particular object. When clicked, this button causes the current object to be set to the object with the ID set by the user in the text input box to the right. In the figure the user has entered the ID 51 (which is the current object). Again, changing the current object causes the video associated with the new current object to be played.

5.5.4 Video-map Navigation Component

Unlike the other components, this component is tailored to one specific use of video. It is used to provide a navigation interface for the video-map instance. Figure 35 shows a screenshot of this component (Figure 35a) and a frame of the video associated with the instance (Figure 35b). In the figure the user is currently at the West Gate and can travel straight ahead or turn 90° clockwise (anti-clockwise turns are not currently supported). This part of the video-map instance can be seen in Appendix D (pp. 283-284, 297-298).



a) Video-map navigation component

b) Image from video-map playback

Figure 35: Video-map Navigation component and video playback image

This component provides the following functionality:

- **Forward**: if this button is not greyed out then pressing it causes the user to travel forwards along the path in front of them. In the figure this causes the user to travel along the path slightly off centre in the image.
- **Rotate clockwise**: if this button is not greyed out then pressing it will cause the user to rotate 90° clockwise at the current junction.
- **Rotate anti-clockwise**: this provides the same functionality as the previous button, but turns the user in the opposite direction (not currently implemented).
- **Speed**: a popup list is used to control the speed of forward travel from stop, through walk, to run. In the figure the user is currently stopped.
- **Location**: the text in the top left hand text box of the interface informs the user of their current location. In this example the current location is West Gate.
- **Heading**: the text displayed in the top middle text box informs the user of their heading within the video-map. As the user moves around the video-map, both this and the Location are updated appropriately. In this example the current heading is WtoN which indicates that the user is turning from facing West to facing North.

5.5.5 Simple Timeline Component

Like the Video-map Navigation component, this component is not essential to the creation and use of instances and schemas. In fact, it is not even shown in Figure 25. It provides a graphical representation of the relative temporal extents of objects in an instance. Many of the systems outlined in Chapter 2, such as Stratagraph [12], provide such timelines to allow a user to obtain an overview of the ways in which segments of an instance overlap in time. In these systems time is represented on the horizontal axis from left to right.

An extract of a screenshot of this component is shown in Figure 36. This displays a timeline in which objects and their temporal extents are represented by rectangles. These rectangles contain the object type followed by the key attribute field. Composition is represented on the vertical axis; objects are composed of those below them. Time is represented on the horizontal axis from left to right. This example shows part of the plot segmentation of *Citizen Kane*. The top rectangle (blank in this figure as it's label is on the far left of the timeline) represents the whole film. The rectangles directly below that represent the Parts; here the Part whose key attribute is Xanadu is shown. The rectangles below this represent the Scenes and Sub-Parts (due to the non-hierarchic nature of the plot segmentation schema) *e.g.* the Scene Thompson talks with Raymond is followed by the Sub-Part Fifth flashback. Below this Sub-Part are the Scenes it is composed of. In this case, only the start of the first Scene Kane destroys Susan's room and picks up paperweight murmuring Rosebud is shown. This component does not directly support any navigation within instances, nor any interactive use of objects such as playing selected ones. Its purpose is to display a temporal overview of objects.

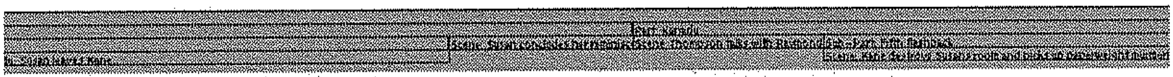


Figure 36: Extracted screenshot of the simple timeline component

As with other User Interface components, this component only communicates with the user and the Application Interface component. It creates the timeline by requesting from the Application Interface component the temporal extent, key attribute, and object type of all objects. It also queries the Application Interface component about the composition relationships between objects in order to the determine objects' positions on the vertical axis.

5.6 Summary

This chapter described the VCMF toolset which supports the creation and manipulation of instances and schemas. It is used in this thesis to support the creation and manipulation of exemplar instances in Chapter 6. These instances are used to demonstrate the validity of the thesis defined in Chapter 3. Moreover, they are used to support investigation of reuse. Therefore, this toolset is important to the thesis as it supports the demonstration and investigations involved. Without the toolset this demonstration would be time consuming, and the investigation of reuse would be difficult to undertake realistically.

Chapter 6 Examples of VCMF's Use

This chapter presents examples of VCMF's use and VCMF's ability to support reuse of video, models, and schemes. Section 6.1 outlines how these examples cover the cross section of video content modelling attributes and purposes discussed in Chapter 2. Furthermore, it outlines how they exhibit the cross section of kinds of reuse outlined in Chapter 2. Section 6.2, Section 6.3, and Section 6.4 detail the three carefully chosen examples, their schemes, models, and reuse exhibited. Section 6.5 summarises the attributes and properties exemplified by the examples.

6.1 Supporting the Thesis

This chapter presents three worked examples of VCMF's use. They purposefully employ all the properties of schemes and models, and illustrate the kinds of reuse of models discussed in Chapter 2. These examples serve two purposes:

1. They demonstrate VCMF's ability to describe a cross section of video content modelling schemes and their models (the primary goal of this thesis).
2. They support the investigation of the feasibility of using VCMF to support reuse of these models, in particular, the reuse of objects from one model in models created from other schemes. This investigation is the subsidiary goal of the thesis.

6.1.1 Demonstrating, Investigating, Evaluating

At this point it is important to distinguish between *demonstration*, *investigation*, and *evaluation*. The demonstration of VCMF's ability to describe a cross section of schemes and their models involves showing that VCMF *can* describe example schemes and models which constitute a cross section of current work; this thesis posits that VCMF can describe the cross section given semantic data modelling's descriptive properties.

This thesis' demonstrative approach is similar to approaches used in other research. For example, establishing HDM's ability to describe applications and classes of applications spanned several years and versions of HDM (see Chapter 3 for a discussion of HDM). Their demonstration involved the use of several examples: 'sales support for a bank in Italy, support for monitoring patient conditions in Germany, and an electronic version of the Gold of Greece exhibit' [35], hypermedia database for technical documentation [72], and an interactive Art Gallery CD-ROM [73]. This does not constitute a *rigorous* demonstration of every aspect of HDM, rather it shows the ability of HDM to support a wide range of applications. Similarly this thesis shows VCMF's ability to support a wide range of applications of video content models and schemes.

Introspective observations on VCMF's use in the demonstration, in particular how well it describes models and schemes, are raised in this chapter and returned to in the next for further discussion. In order to rigorously evaluate how well VCMF describes schemes and models an evaluation needs to be able to separate schemes and models from their representations (the graphical notation and the toolset's representation in this thesis) and evaluate each individually. This is premature at the current stage; first the utility of the approach needs to be demonstrated (undertaken by this thesis), then its components can be evaluated individually.

A feasibility investigation aims to investigate the possibility of something being done and identify problems with attempting to do something. In this thesis the feasibility of using VCMF to support reuse is investigated as it is not clear how well semantic data modelling approaches support reuse. This is accomplished by investigating VCMF's support for different kinds of reuse identified in a taxonomy of reuse. The result of this investigation is a set of problems and successes with using the approach to support reuse. The kinds of reuse which are problematic then form the basis for future research into supporting such reuse. In this thesis issues are identified by introspection; trying to carry out different kinds of reuse and noting the difficulties encountered. Note that the utility of reuse itself is not investigated in this thesis; work discussed in Chapter 2 highlights the need to reuse video and its associated models, Chapter 8 discusses it desirability.

6.1.2 Selecting Appropriate Examples

The aim of the thesis is to show that one framework can describe a cross section of schemes and to investigate its support for reuse. Three examples applications of video content modelling are selected in this chapter; an application is one particular use of one or more schemes. The chosen examples represent a cross section of schemes. Moreover, they exhibit reuse of parts of models from one application in another. In Chapter 2 each system typically supported one application and did not support reuse of video and associated annotations by reference between applications. In order to reduce the time spent constructing these examples a criteria for selection is that plenty of analyses and materials must be readily available. The thesis is not concerned with examining how these analyses are carried out, nor how raw are materials collected; it is concerned with how analyses of raw materials can be represented, and how these representations can be reused.

The chosen examples show models which have different primary purposes: analysis or presentation. Clearly video is presented to the user for both purposes; the distinction between the terms is used to indicate whether the scheme is intended to primarily support the analysis of video content through the construction of models (*e.g.* style analysis) or the construction of models which support the presentation of video segments (*e.g.* the video-map). This distinction is discussed in greater detail in Chapter 2.

VCMF is a framework for modelling the data and structure of video content models; it is not primarily interested in the *purpose* of models. The next chapter uses experiences from this chapter to draw out the implications of different purposes of models on their schemes. The fact that VCMF is aimed at describing a cross section of schemes, not a cross section of purposes is especially pertinent to the third example (presented in Section 6.4). This example's domain of use and its purpose are similar to the first example's domain of use and its purpose (presented in Section 6.2) but the schemes used are markedly different.

Table 12 illustrates the sum of schemes' characteristics (and hence model characteristics) exhibited by the examples selected to demonstrate the thesis (each example uses multiple schemes). Chapter 2 discusses these characteristics with reference to current literature. To recap, they are evident in the current literature, and constitute a cross section of characteristics of schemes in the literature. The table illustrates how, although there are common characteristics between the examples, some characteristics are unique to one scheme alone. The ability of VCMF to describe this cross section of schemes and their models supports the thesis (see Chapter 7 for discussion). These characteristics do not include the different kinds of annotations discussed in Chapter 2 *e.g.* free text, representative frames. This is because VCMF objects are typed, and attribute values are defined using BNF, this means that objects can represent typed annotations containing unstructured data, structured data, and structured collections of data (see Chapter 4). These characteristics also do not include the different forms of segmentation (segments, clips, or documents) as VCMF uses segments which subsume other forms of segmentation (see Chapter 2 for a definition of segments, and Chapters 7 and 8 for a discussion of their use).

6.1.3 Reuse Exhibited by the Examples

Table 13 presents a taxonomy of the reuse of objects exhibited by the examples in this chapter and used to investigate VCMF's support for reuse. The taxonomy itself is derived from the work of Garzotto *et al.* [74] who discuss reuse of information in multimedia applications. It shows how the examples selected together illustrate each form of object reuse at least once (except reuse of objects in a different application, using the same scheme). Reuse is defined in Chapter 2 as being the use, either by copying or reference, of something in a different context. The new context may be within the same application. For example, the plot analysis application uses a scheme to analyse the structure of a film's plot, and another to query the model of the plot's structure. Alternatively, the new context may be in a different application in which case it might be reused in a different scheme from its original use. This chapter focuses on the reuse of objects. Other forms of reuse investigated in this chapter are the reuse of schemes (returned to in Chapters 7 and 8) and the reuse of video itself (without any models). These forms of reuse are investigated less thoroughly than reuse of objects as reuse of objects (especially between schemes) is identified in Chapter 3 as a key issue in video content modelling.

Reuse of objects in VCMF is supported by referencing them in their new context. As discussed in Chapter 2, this is preferable to reuse by copying as: video is not duplicated, annotations remain with their segments, and annotations do not become inconsistent.

			Plot segmentation Section 6.2	Video-map Section 6.3	Style analysis Section 6.4
Purpose	Analysis		√		√
	Presentation	Deterministic	√		
		Non-deterministic		√	
Characteristics	Collection	Homogeneous	√	√	
		Heterogeneous	√	√	√
	Membership relationships	1-m	√		√
		1-x		√	√
	Sequential relationships	None		√	√
		Arbitrary	√	√	
		Inferred	√		
		Predefined			√
		Temporal			√
	Schema specific relationships			√	√

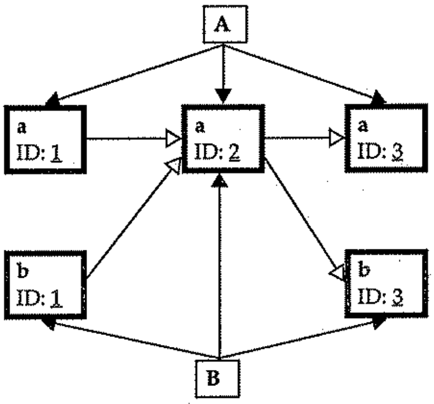
Table 12: Purposes and characteristics shown in examples

Figure 37 illustrates a simple example of VCMF's support for reuse by reference. In the figure object a with ID: 2 (object a2 for short) is a member of collection A, and is reused in collection B. The figure illustrates the relationships which support this reuse and ensure that the correct sequences of member objects are preserved: for collection A the sequence of member objects is a1 → a2 → a3, and for collection B the sequence of member objects is b1 → a2 → b3. Systems such as Sawhney *et al.*'s HyperCafé [173][174] also support reuse in this manner. To avoid confusion during such reuse VCMF assumes that linear sequences only hold within a collection. This means that VCMF assumes that there are only two linear sequences in the example (a1 → a2 → a3, and b1 → a2 → b3), rather than the four possible (a1 → a2 → a3, b1 → a2 → b3, a1 → a2 → b3, and b1 → a2 → a3).

Especially important in Table 13 is the reuse of objects from one model in a model of a different scheme, used for a different application. This important kind of reuse is

		Scheme	
		Same scheme	**Different scheme**
Application	**Same application**	Reuse of Shots from the narrative of *Citizen Kane* in its end credits.	The guided tour of a video-map. Kane's life viewed through flashbacks in chronological order. Comparisons between patterns of use in style analysis and the plot segmentation.
	Different application	Documentary about *Citizen Kane*. (unsuccessful - schema had to be modified, see Section 6.2)	Documentary about the area represented in a video-map. Comparisons between patterns of use in style analysis and the plot segmentation.

Table 13: Kinds of object reuse exhibited in the chosen examples



Relationship types	a ──▶ b	a ──▷ b
Name	*a* Contains *b*	*a* Precedes *b*
Inverse name	*b* is_within *a*	*b* follows *a*

Figure 37: Simple example of reuse

emphasised in the table by a double border and is successfully supported by VCMF. However, the reuse of objects in a different application, but using their original scheme is not supported in this example as the schema has to be slightly modified for the new application. Similarly, the plot segmentation schema has to be modified to support the reuse of objects from *Citizen Kane* in the end credits of the film. These problems are discussed in this chapter, and returned to in the discussions of Chapter 7. They do not invalidate the thesis as its aim is to show that a semantic data modelling approach (VCMF in this case) can be used to describe a cross section of schemes, and to investigate the reuse of their models. As discussed previously, an investigation's role is to discover problems with an approach. Therefore the investigation has fulfilled its role by identifying these problems.

6.1.4 Structure of This Chapter

The examples in this chapter are presented in the order in which they were constructed. The first example was used to finalise the development of the implemented system and to provide a strong central example around which the other examples fit. The other two examples were selected to exemplify properties of schemes and reuse not illustrated in the other examples. This made selecting the third example troublesome as it needed to exhibit properties not found in the other two examples.

The description of examples in the following sections all follow the same format. First, the purpose of the example is presented, followed by a description of its domain of use. A domain specific description of the example scheme is then given, and a VCMF schema which represents it is presented. These are followed by the domain based description of the example model and a VCMF instance which represents it. Reuse of the instance and possibly the schema is then discussed, after which there is a short summary of the properties demonstrated.

6.2 Plot Analysis of Narrative Films

6.2.1 Purpose of This Example

This example provides a strong central example around which the other two examples fit to provide a cross section of schemes.

6.2.2 Domain of Plot Analysis

The scheme illustrated in this example is designed to support the analysis of plot in narrative films. Bordwell *et al.* define plot as:

> "**plot**: In a narrative film, all the events that are directly presented to us, including their causal relations, chronological order, duration, frequency, and spatial locations. Opposed to *story*, which is the viewer's imaginary construction of all the events in the narrative." [21] (their emphasis)

Non-narrative films, which are not plot based, require different analysis techniques and so are not considered in this example.

Such plot analysis is not the preserve of film academics; many people critically examine films including TV presenters, newspaper reporters, and even the general public in everyday conversation. Someone analysing a film tries to gain a better understanding of the film, how it effects them, and how it creates these effects; this understanding can then be used as the basis of discussion, review, and so on. Academics seek a more formal examination of a film and its effects, consequently a large amount of literature has developed which describes how to critically analyse films from a formal academic perspective [135]. The large amount of readily available literature and source materials (video recordings and transcriptions of these recordings) make plot analysis ideal as the strong central example of VCMF's use.

Plot segmentation is the first part of any academic analysis of a narrative film. This involves the analysis of the plot in terms of its sequence and composition (non-narrative films, which are not story based, require different analysis techniques).

6.2.3 Plot Segmentation Scheme and Schema

The general structure of plot segmentations of narrative films discussed by Bordwell *et al.* [21] is illustrated in Figure 38. Italic text in the figure describes the relationships between the segments (described by non-italic text). The general structure of plot segmentations is as follows:

- **Parts** represent major divisions in the plot, and are composed of sequences of Sub-Parts and Scenes.
- **Sub-Parts** represent logical divisions of the plot within Parts, and are composed of sequences of Scenes.
- **Scenes** represent distinct phases of action occurring within a relatively unified space and time (relative to the plot of the whole film), and are composed of sequences of Sequences and Shots.
- **Sequences** represent logical divisions within Scenes, and are composed of sequences of Shots.
- **Shots** are uninterrupted sequences of video from one camera and form the lowest level of granularity of the segmentation.

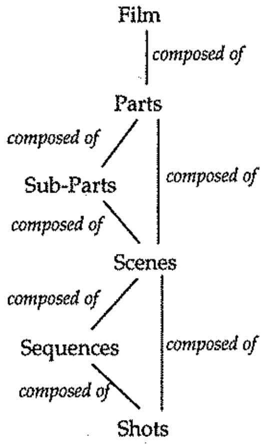


Figure 38: General structure of plot segmentations discussed by Bordwell *et al.* [21]

In this thesis' terminology, this description of the general structure of plot segmentations constitutes a rough outline of a video content modelling scheme.

6.2.3a Plot segmentation schema

A VCMF schema (the plot segmentation schema) representing the general structure of plot segmentations is illustrated in Figure 39 and detailed in Appendix E (pp. 310, 314).

In terms of VCMF object types, Shots constitute Units as they are the lowest level of granularity of the scheme, all other object types are Attribute-Ordered Collections as their member objects will be linearly ordered in terms of start_frames in models of plot segmentations. Additionally, the schema specifies that the objects of an instance of it have the following attributes:

- **Films**: name, and bibliography.
- **Parts**: description, and unique identifier which is either C (beginning credits), E (End credits), or a number. The format of identifiers is taken from Bordwell *et al.* [21].
- **Sub-Parts**: description, and type (currently "Flashback" or "Other").
- **Scenes**: description, and unique identifier which is made up of the identifier of the Part they are a member of (possibly indirectly via a Sub-Part) and a lower case letter.
- **Sequences**: description, and unique identifier constructed from the identifier of the Scene they are a member of followed by an upper case letter.
- **Shots**: description, and list of characters appearing in the Shot.

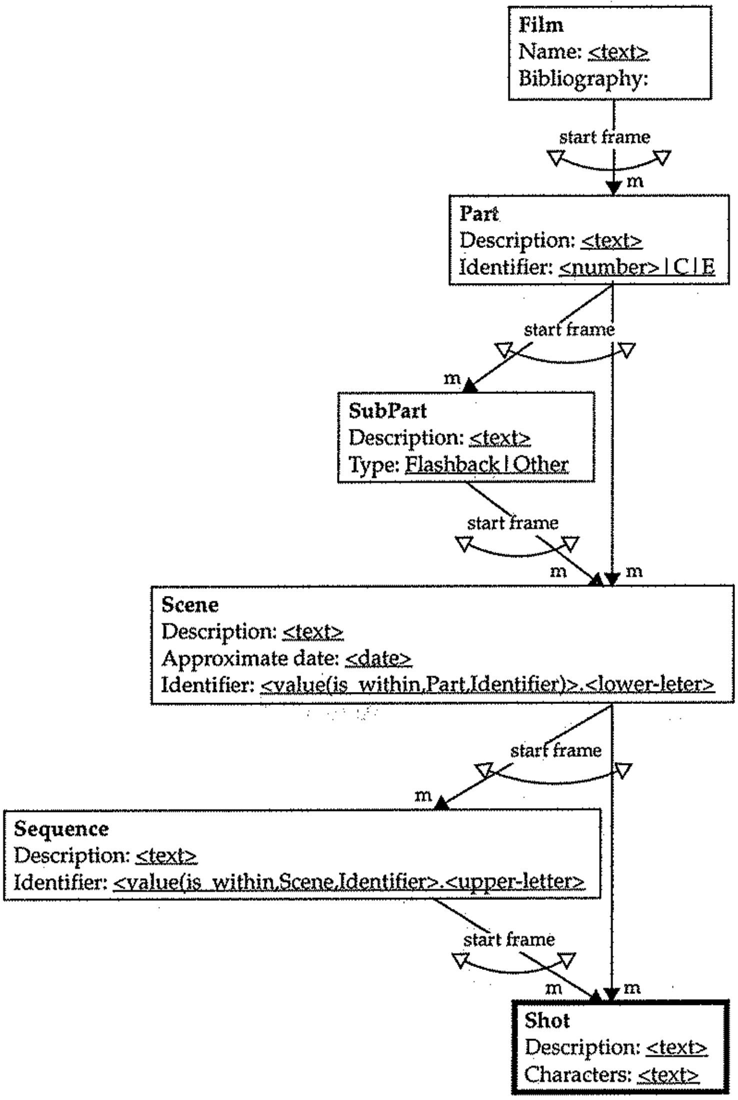
6.2.4 Plot Segmentation Model and Instance

The film *Citizen Kane*, directed by Orson Welles, is used as the example plot analysis in this thesis. Once again, this is because it has been extensively studied by film analysts such as Bordwell *et al.* [21] and Kael *et al.* [104], and so can provide a strong example of the use of VCMF. Moreover, it illustrates several examples of possible reuse of video (illustrated and discussed later in this section).

Citizen Kane unconventionally starts with the death of the main character, a once rich and powerful newspaper tycoon, Charles Foster Kane (played by Orson Welles), owner of the Inquirer newspaper. The rest of the film follows the journalist Thompson's investigation of Kane's life to try and uncover the secret of Kane's dying word: "Rosebud". The reporter visits Kane's wives, friends, and business associates to help uncover the mystery, but never reaches a satisfactory conclusion. At the time of the film's release (1941) it caused great controversy due to its real-life correspondences to the newspaper publisher William Randolph Hearst who tried to get the film banned and ensured that all his papers gave it poor reviews.

Table 14 is Bordwell *et al.*'s [21] outline plot segmentation of *Citizen Kane* structured according their general structure illustrated in Figure 38. This plot segmentation is only concerned with Parts, Sub-Parts and Scenes as this level of segmentation is typically used by analysts to provide an overview of the film's plot. Further details of Sequences and Shots within Scenes are constructed in an *ad-hoc* fashion for particularly interesting aspects of the plot. Table 15 is Bordwell *et al.*'s detail of the Sequences in Scene 2.a ("News on the March"). This Scene is a fictional newsreel describing Kane's life; it sets the stage for the film, and acts as a rough outline of the rest of the film

The plot segmentation itself contains *Flashbacks* which are represented as Sub-Parts; flashbacks present events before the main period of the story. In *Citizen Kane* the main



Relationship types	a ——▶ b	a ——▷ b
Name	*a* Contains *b*	*a* Precedes *b*
Inverse name	*b* is_within *a*	*b* follows *a*

Figure 39: Schema representing the structure of plot segmentations

period of the story is a reporter's investigation into Kane's life and so flashbacks are used to present various character's memories of Kane during his life. For example, the first flashback (in Part 4) presents Thatcher's (Kane's guardian) memoirs of Kane's life from boyhood through to the collapse of his newspaper chain

The text in Table 14 and Table 15 is essentially the annotations of segments in a video content model representing the plot segmentation of the film *Citizen Kane*. The horizontal layout of these tables indicates the composition of these annotations. Annotations on the left are composed of those on the right. For example, Part 2 ("Projection room") is composed of Scenes 2.a ("News on the March") and 2.b ("Reporters discuss 'Rosebud'").

Part	Sub-Part	Scene
C. Credit title		
1. Xanadu: Kane dies		
2. Projection room:		
		a. "News on the March"
		b. Reporters discuss "Rosebud"
3. El Rancho nightclub: Thompson tries to interview Susan		
4. Thatcher library:		
		a. Thompson enters and reads Thatcher's manuscript
	First flashback	b. Kane's mother sends the boy off with Thatcher
		c. Kane grows up and buys the *Inquirer*
		d. Kane launches the *Inquirer's* attack on big business
		e. The Depression: Kane sells Thatcher his newspaper chain
		f. Thompson leaves library
5. Bernstein's office:		
		a. Thompson visits Bernstein
	Second flashback	b. Kane takes over *Inquirer*
		c. Montage: the *Inquirer's* growth
		d. Party: the *Inquirer* celebrates getting the *Chronicle* staff
		e. Leland and Bernstein discuss Kane's trip abroad
		f. Kane returns with his fiancee Emily
		g. Bernstein concludes his reminiscence
6. Nursing home:		
		a. Thompson talks with Leland
	Third flashback	b. Breakfast table montage: Kane's marriage deteriorates
		c. Leland continues his recollections
	Third flashback (continued)	d. Kane meets Susan and goes to her room
		e. Kane political campaign culminates in his speech
		f. Kane confronts Gettys, Emily, and Susan
		g. Kane loses election and Leland asks to be transferred
		h. Kane marries Susan
		i. Susan's opera premiere
		j. Because Leland is drunk, Kane finishes Leland's review
		k. Leland concludes his reminiscence
7. El Rancho nightclub:		
		a. Thompson talks with Susan
	Fourth flashback	b. Susan rehearses her singing
		c. Susan's opera premiere
		d. Kane insists that Susan goes on singing
		e. Montage: Susan's opera career
		f. Susan attempts suicide and Kane promises she can quit singing
		g. Xanadu: Susan bored
		h. Montage: Susan plays with jigsaw puzzle
		i. Xanadu: Kane promises picnic
		j. Picnic: Kane slaps Susan
		k. Xanadu: Susan leaves Kane
		l. Susan concludes her reminiscence
8. Xanadu:		
		a. Thompson talks with Raymond
	Fifth flashback	b. Kane destroys Susan's room and picks up paperweight, murmuring "Rosebud"
		c. Raymod concludes his reminiscence: Thompson talks with the other reporters; all leave
		d. Survey of Kane's possessions leads to a revelation of Rosebud; exterior of gate and of castle; the end
E. End Credits		

Table 14: Plot segmentation of *Citizen Kane* from Bordwell *et al.* [21]

Similarly, Part 5 ("Bernstein's office") is composed of Scene 5.a ("Thompson visits Bernstein"), the "Second flashback" Sub-Part, and Scene 5.g ("Bernstein concludes his reminiscence"). The "Second flashback" Sub-Part is in turn composed of Scenes 5.b, 5.c, 5.d, 5.e, and 5.f. The vertical sequence (top to bottom) of lines of text in the tables represents the linear ordering of annotations within the model. For example, Part 1 is before Part 2, Scene 4.f is before Scene 5.a and Part 5.

Scene		Sequence	
2a.	"News on the March"	A.	Shots of Xanadu
		B.	Funeral: headlines announcing Kane's death
		C.	Growth of financial empire
		D.	Silver mine and Mrs. Kane's boarding house
		E.	Thatcher testimony at congressional committee
		F.	Political career
		G.	Private life; weddings, divorces
		H.	Opera house and Xanadu
		I.	Political campaign
		J.	Depression
		K.	1935: Kane's old age
		L.	Isolation at Xanadu
		M.	Death announced

Table 15: Sequences in the "News on the March" Scene from the plot segmentation of *Citizen Kane* [21]

A system such as VideoSTAR [92] is ideally suited to storing and representing such plot segmentations as well as relating these annotations to their associated video (though the structure of plot segmentations presented here is more complex than the structure supported by VideoSTAR). Chapter 2 provides a full discussion VideoSTAR's representation of video content using hierarchies of annotations. It also highlights the high proportion of systems which describe such video content models. The fact that there is such a high proportion of work in this area reinforces the contention that this example is a strong central example of the use of video content models. Of course, these systems could represent this plot segmentation, but they would be unable to adequately describe the other examples in this chapter; this chapter demonstrates that VCMF can.

6.2.4a The Citizen Kane instance

Appendix D (pp. 278-283) details the instance (*Citizen Kane* instance) representing *Citizen Kane*'s plot segmentation. Figure 40 is a recreated extract from this instance which illustrates how the types of the schema have been instantiated as objects and relationships with attributes assigned appropriate values. In keeping with the VCMF approach, the instance represents video segments and annotations as objects which are related to each other by relationships (the dotted lines indicate that further objects and relationships exist but are not shown in the diagram). The annotations and relationships are taken from Table 14 and Table 15. The video itself is a digitised copy of *Citizen Kane* (frame rate: 8 frames per second, frame size: 192 by 144 pixels, total size: 265.1 Mb). The diagram in Appendix D was produced using the VCMF toolset described in Chapter 5. The *Citizen Kane* instance contains 155 objects and 132 relationships.

6.2.5 Reuse of Citizen Kane

Three examples of reuse of objects and their associated video are discussed in this sub section. The first involves querying the *Citizen Kane* instance to create a new collection, the second involves manually including objects from the instance in another instance, the

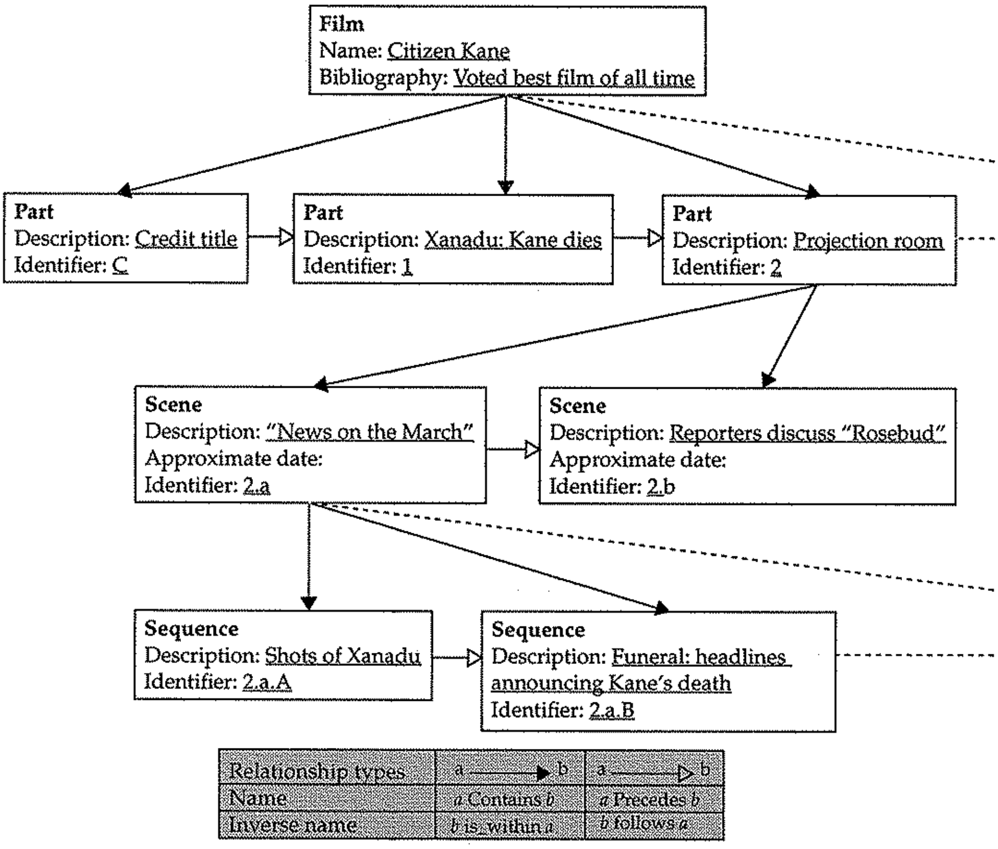


Figure 40: Extract from the instance representing the plot segmentation of *Citizen Kane*

third involves the reuse of Shot objects from the main narrative of *Citizen Kane* in its end credits.

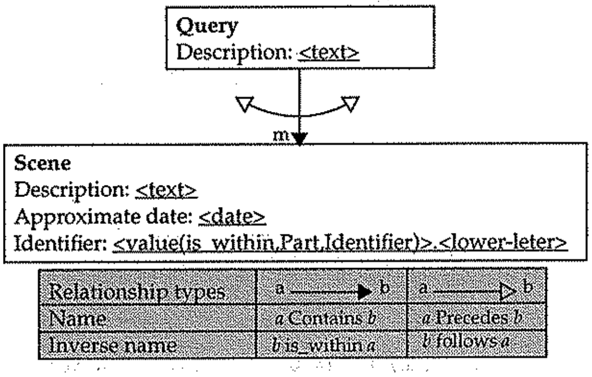
6.2.5a Reuse of flashbacks

Most of *Citizen Kane* consists of *flashbacks* referring to Kane's life. A flashback is an 'alteration to the story order in which the plot moves back to show events that have taken place earlier than the one already shown' [21]. Moreover, in *Citizen Kane,* these flashbacks are not presented in the chronological order of Kane's life. They form part of an investigative journalist's exploration of Kane's life through recollections by Kane's friends and family.

An interesting reuse of objects of the *Citizen Kane* instance is the creation of a new sequence presenting Kane's life in chronological order. Because each Scene object has an attribute for the approximate date it represents (each Scene presents only one part of Kane's life) the VCMF toolset can create a collection representing Kane's life in chronological order with the following SVQL query (assuming the schema illustrated in Figure 41 exists):

```
select from Scene into Query where Approximate date > 0 order by Approximate date
```

For clarity the resultant Query object and its relationships to objects of the *Citizen*



Relationship types	a ——▶ b	a ——▷ b
Name	*a* Contains *b*	*a* Precedes *b*
Inverse name	*b* is_within *a*	*b* follows *a*

Figure 41: Schema for query of *Citizen Kane*

Kane instance are not illustrated in the diagram in Appendix D. Figure 42 shows the first 5 objects in the Collection created by this query (there are 26 objects in total). In this figure only the Query and Scene objects and their relationships are shown; other objects and relationships do exist but are not shown for the sake of clarity. The Query object shown in the diagram has had its description attribute instantiated as Kane's life in chronological order to describe its contents. Note that by default this description attribute is instantiated with the query text itself. The extract shows how relationships have been instantiated to support reuse of the objects by reference. The Scene objects can now be viewed in the chronological order of Kane's life, or in the order dictated by the plot segmentation instance.

6.2.5b Reuse in a documentary about Citizen Kane

The second example of reuse of objects from this instance involves including objects in a different instance created using the plot segmentation schema. The *50th Anniversary* edition of the video cassette of *Citizen Kane* includes an additional documentary *Reflections on Citizen Kane* in which various film directors explain their favourite moments from the film. To help illustrate their favourite parts of the film, footage from the film is included in the documentary.

The original intention of this example was to use the plot segmentation schema to create an instance representing the documentary. This would have shown reuse of objects from one instance in the same schema, but for a different application, and the reuse of a schema. However, this causes problems as the plot segmentation schema employs Attribute-Ordered Collections which are ordered on the start_frame. Reusing objects from the *Citizen Kane* instance in the documentary instance would mean that ordering members of Collections by their start_frames would not produce the desired sequences. This is because objects in the documentary instance are stored in wholly different parts of the video storage device in comparison to the *Citizen Kane* instance. To solve this problem, the plot segmentation schema was adjusted; each Attribute-Ordered Collection

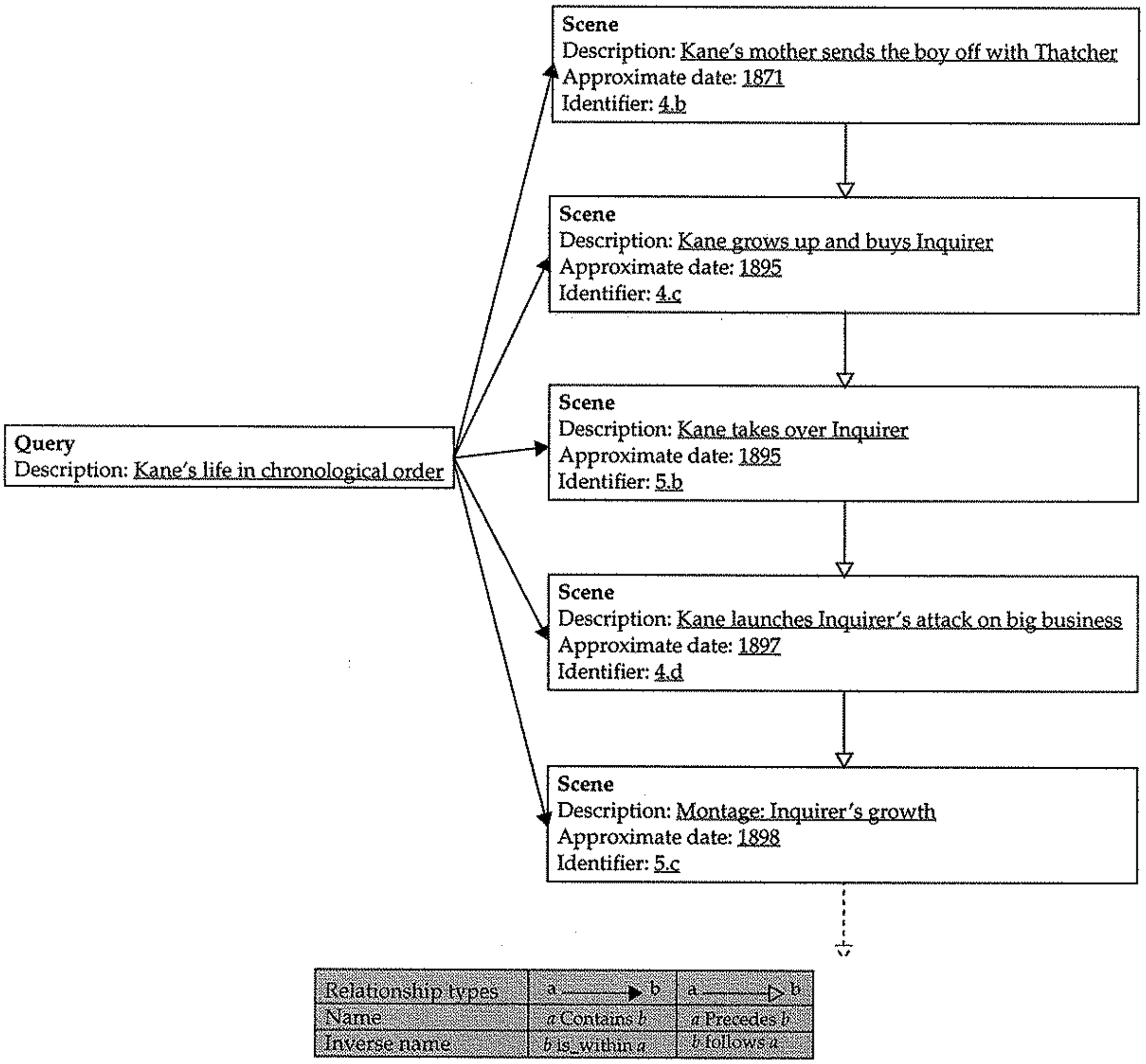


Figure 42: Extract of result of query

was replaced by a User-Ordered Collection as shown in Figure 43. This highlights one of the major problems encountered when reusing schemas designed for analysis for presentation (further discussed in Chapter 7): schemas used for analysis (*e.g.* plot segmentation) are typically concerned with the sequence of objects in the video recording whereas schemas used for presentation (*e.g.* constructing the documentary) are concerned with creating new sequences of objects.

Although most documentaries employ a categorical form as opposed to the narrative form used in plot segmentation, the structural semantics (the way in which segments are composed) and naming conventions are the same. The differences lie in the semantics of the segments. For example, in categorical form, 'Parts treat distinct parts of some subject' [21] rather than major divisions in the plot (of a narrative film). Likewise, in a categorical form, Sub-Parts and Scenes represent divisions within parts of a subject rather than parts

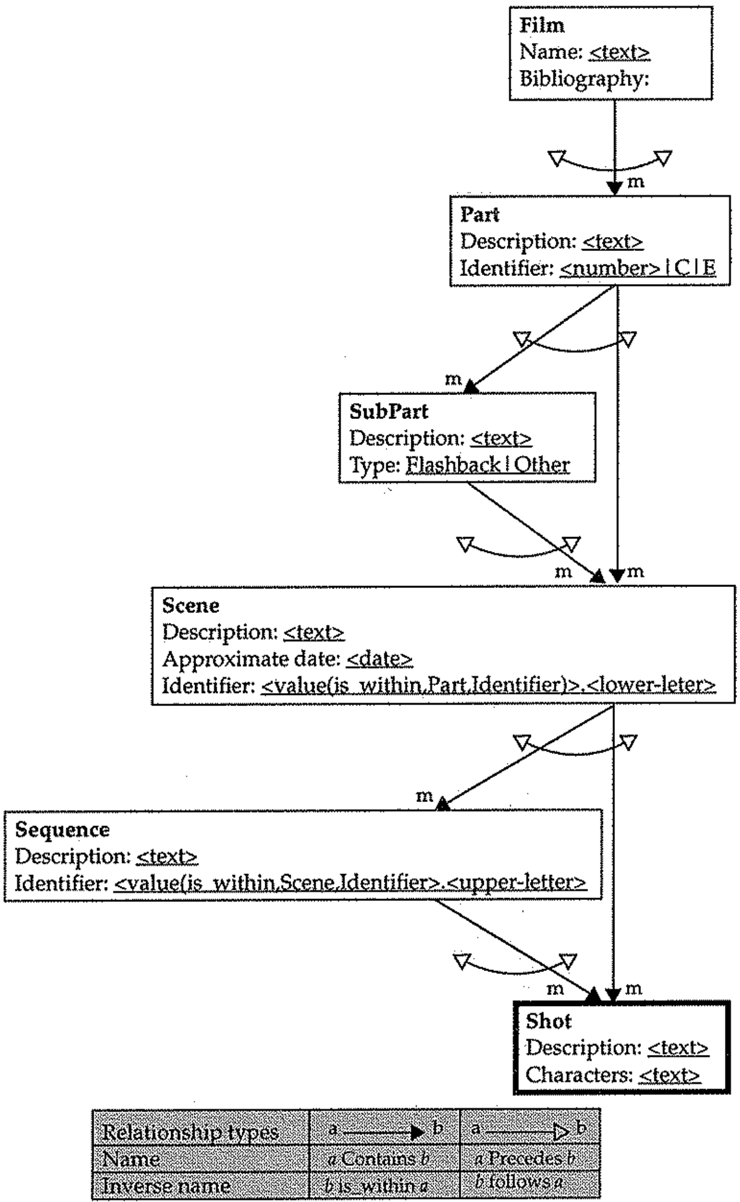


Figure 43: Modified plot segmentation schema adjusted for documentary construction

of a plot. Because of this similarity in structure and naming, the plot segmentation schema is used for the construction of the documentary with the slight modifications discussed in the previous paragraph.

The VCMF toolset and the modified plot segmentation schema were used to construct an instance which represents the documentary (the documentary instance). Footage digitised from the video cassette of the documentary was mixed with objects from the *Citizen Kane* instance where appropriate *i.e.* the *Citizen Kane* footage in the documentary instance was replaced by references to objects in the *Citizen Kane* instance.

The whole documentary instance is illustrated in Appendix D (pp. 292-296). It should be noted that for brevity only the first 3 Parts are fully segmented into their Scenes, Shots *etc.* The other Parts are not segmented as the aim of this reuse example is to investigate such reuse which does not necessitate recreating the whole documentary.

Figure 44 illustrates an extract showing the reuse of a single Shot from the *Citizen Kane* instance in the documentary instance (shown in full in Appendix D). The documentary instance takes up the bottom half of the picture and is related by reuse to the *Citizen Kane* instance in the top half. The Shot of Kane gasping "Rosebud" on his deathbed is referenced in the documentary as part of the description of Welles' life. In the figure the thick lines illustrate the relationships which make this reuse possible. The dotted lines indicates that more of the instance exists than is shown.

This reuse differs from the other reuse example because objects from the *Citizen Kane* instance are mixed with objects of the documentary instance which is used for a different application.

6.2.5c Reuse in the end credits

The third example of reuse involves reusing objects from *Citizen Kane*'s narrative in its end credits. In the end credits the main characters of the film are shown along with the actors' names. Table 16 illustrates the plot segmentation adapted from the shooting script [104] for this end Scene along with the Scenes that the Shots are reused from. The Scene starts with a title caption explaining that most of the actors are new to the film industry. It then reminds the viewers of the characters by showing short Shots of them from the narrative, overlaid with the actor's name.

The VCMF instance supporting the reuse in the end credits can be seen in Appendix D (pp. 278-283). This reuse poses three problems which are discussed in full in Chapter 7:

1. The reused footage is only part of the Shot from the narrative. In the VCMF supported reuse the whole Shot is reused which can sometimes be much longer than required.
2. In the end credits the actors' names are overlaid on top of the reused footage. VCMF currently does not support such overlaying of one piece of video (the actor's name in this case) on another (the Shot of the actor in this case). Therefore in the instance there is no overlay of actors' names with their characters.
3. The plot segmentation schema used for the rest of *Citizen Kane* cannot be used to support the modelling of the end credits. This is because the Scene collection members are attribute ordered on their start_frame, but the Shots reused in the end credits are not in such a sequence (as illustrated in Table 16); they must be manually inserted into the End Credits Scene Collection. To support this, another modification of the plot segmentation schema is needed; this is illustrated in Figure 45. This modification means that there is now a Contains relationship from the Scene Collection to the Shot Unit, but unlike the other composition relationships in this schema, this relationship is *user ordered.*

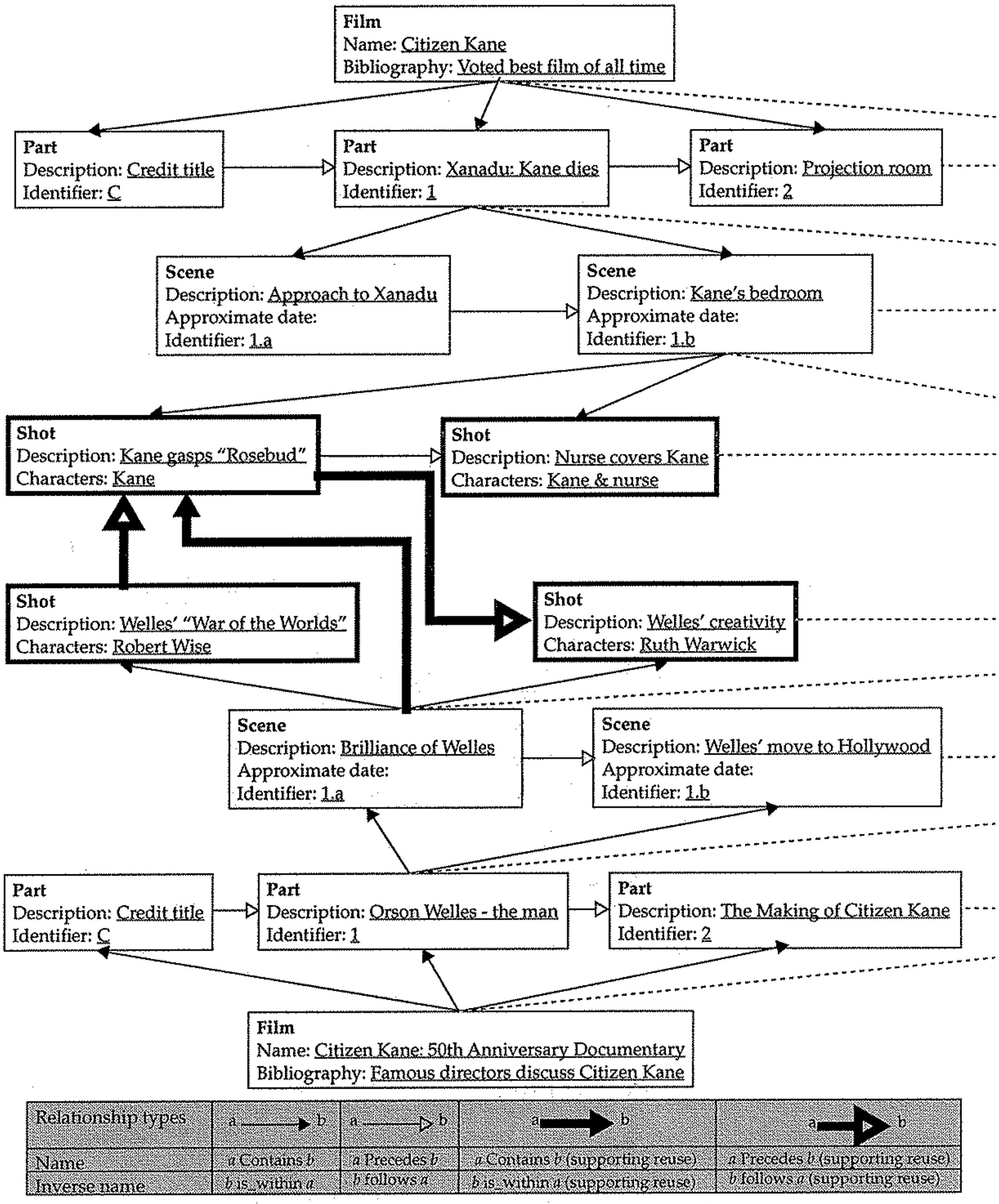

Film
Name: Citizen Kane
Bibliography: Voted best film of all time
Part
Description: Credit title
Identifier: C
Part
Description: Xanadu: Kane dies
Identifier: 1
Part
Description: Projection room
Identifier: 2
Scene
Description: Approach to Xanadu
Approximate date:
Identifier: 1.a
Scene
Description: Kane's bedroom
Approximate date:
Identifier: 1.b
Shot
Description: Kane gasps "Rosebud"
Characters: Kane
Shot
Description: Nurse covers Kane
Characters: Kane & nurse
Shot
Description: Welles' "War of the Worlds"
Characters: Robert Wise
Shot
Description: Welles' creativity
Characters: Ruth Warwick
Scene
Description: Brilliance of Welles
Approximate date:
Identifier: 1.a
Scene
Description: Welles' move to Hollywood
Approximate date:
Identifier: 1.b
Part
Description: Credit title
Identifier: C
Part
Description: Orson Welles - the man
Identifier: 1
Part
Description: The Making of Citizen Kane
Identifier: 2
Film
Name: Citizen Kane: 50th Anniversary Documentary
Bibliography: Famous directors discuss Citizen Kane
Relationship types
a b
Name
Inverse name
a Contains b
b is_within a
a Precedes b
b follows a
a Contains b (supporting reuse)
b is_within a (supporting reuse)
a Precedes b (supporting reuse)
b follows a (supporting reuse)

Figure 44: Reuse of one Shot in the documentary *Reflections on Citizen Kane*

6.2.6 Summary: Properties Exhibited in the Plot Analysis Example

This first example shows the use of a VCMF schema to describe a scheme which is used for analysis and presentation (deterministic), and has the following structural properties (with respect to Table 12 and the characteristics identified in Chapter 2):

Scene	Shot	Scene reused from
E.a End	Title: Most actors are new to motion pictures	
	Leland: Joseph Cotten	6.a
	Susan: Dorothy Comingore	7.a
	Kane's mother: Agnes Moorehead	4.b
	Emily: Ruth Warwick	6.b
	Gettys: Ray Collins	6.f
	Carter: Erskine Sanford	5.b
	Bernstein: Everett Sloane & Thompson: William Alland	5.a
	Raymond: Paul Stewart	8.c
	Thatcher: George Coulouris	4.c
	Title: Credits	

Table 16: End Credit Scene adapted from the shooting script [104]

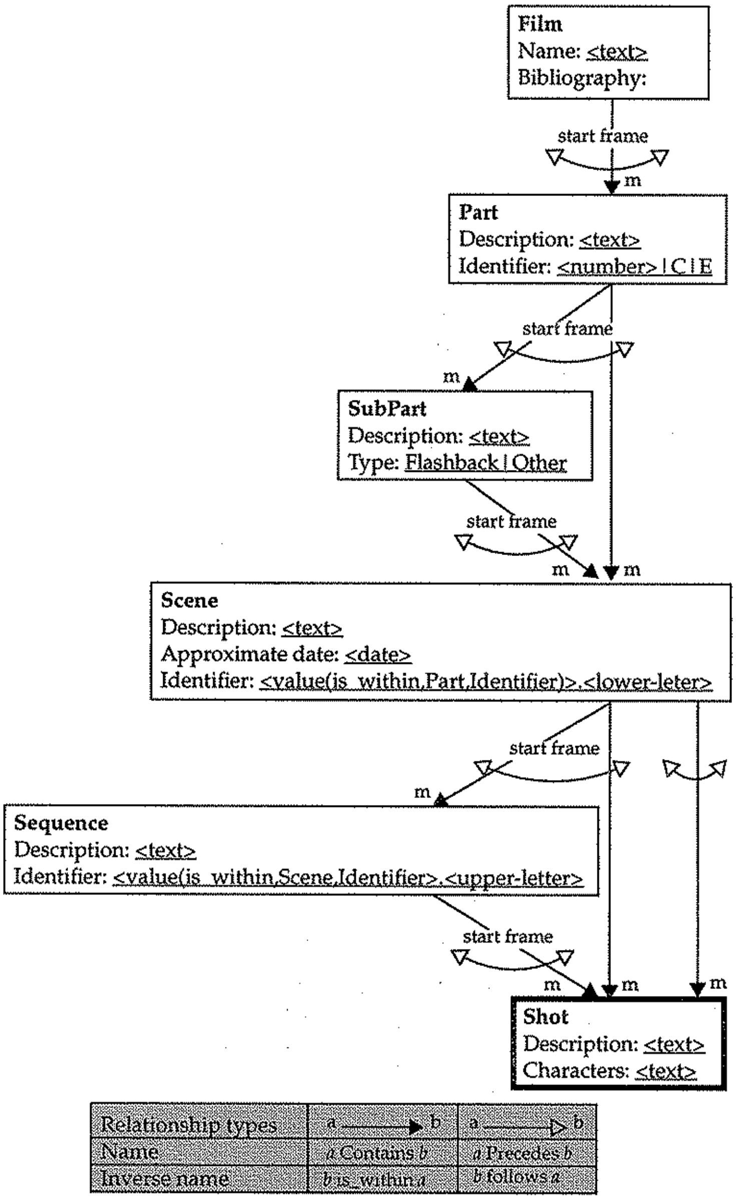


Figure 45: Modified plot segmentation schema to support reuse in end credits

1. **Homogeneous collections**

   Object types Film, Sub-Part and Sequence have homogeneous members.

2. **Heterogeneous collections**

   Object types Part and Scene have heterogeneous members.

3. **1-m membership relationships**

   All Collections are related to their member objects by relationships with 1-m cardinality.

4. **Arbitrary sequences of member objects**

   The Query Collection, and the Collections of the modified plot segmentation schema all arbitrarily order their members.

5. **Inferred sequences of member objects**

   The Collections of the original plot segmentation schema all order their members by their start_frames.

The reuse in this example considers VCMF's ability to support:

1. **Reuse of objects and their associated video in the same application (different schema)**

   Exemplified by the Query Collection containing Kane's life in chronological order.

2. **Reuse of objects and their associated video in a different application (different schema)**

   Objects from the *Citizen Kane* instance are reused in the documentary instance. This reuse highlighted the difficulties of reusing a schema designed for analysis to construct presentations; the initial intention was to reuse the plot segmentation schema to construct the documentary instance. This is further discussed in Chapters 7 and 8.

3. **Reuse of objects and their associated video in the same application (same schema)**

   Objects from the narrative of *Citizen Kane* are reused in its end credits. However, there are three problems associated with this reuse. This means that the end credits created from reuse are not the same as the original end credits. Again, this is further discussed in Chapter 7.

In summary, this first example demonstrates that VCMF can describe the scheme and models of plot segmentation which is a salient use of video content models. This is the first step in demonstrating VCMF's ability to model a cross section of schemes. Moreover it is the first step in investigating VCMF's support for reuse; several problems and successes with reuse have been identified. The next sections describe examples of VCMF's use which exhibit other properties of video content models discussed in Chapter 2.

6.3 Video-Map

6.3.1 Purpose of the Video-Map Example

This example demonstrates the use of VCMF to describe relationships which are not Membership, Sequential, or Temporal: Schema specific relationships. It also shows two kinds of reuse: the reuse of objects within the same application, and the reuse of objects in a different application using a different schema. The latter is an important kind of reuse as it shows that one of the main aims of VCMF is achievable: the reuse of objects in instances of different schemas.

Furthermore, this example demonstrates VCMF's ability to describe a scheme and model which are used for non-deterministic presentation. That is, the sequence in which objects of an instance are presented to the viewer is not known beforehand. Indeed, it is the viewer who interactively influences the sequence at run time. This example is typical of uses of video in the new paradigm discussed in Chapter 2 such as Davenport *et al.*'s ConText [53] which allows a viewer to influence a documentary as it is presented.

6.3.2 Domain of Example

Video-maps have been discussed for many years from Lippman [121] in the 1980s through to recent work by Sawhney [174] in 1997. Basically, a video-map provides a user with an interactive *virtual journey* through a space. The user is presented with video playback of recordings of journeys through a space and can influence the system's next choice of video to display. Typically there are decision points at which the user can elect to view video footage of different journeys from that point. The construction of video-maps often involves a large amount of footage capture as each journey needs to be traversed, and each possible turn needs to be captured.

Sawhney's work is set in a café space in which the user navigates around the café and can elect to hold interactive conversations with people in the café. These conversations are video recordings of people talking; at certain points the user can influence the conversation by choosing the next video recording to be displayed.

Lippman's system provides the user with a virtual car journey through a town. The user can adjust the speed of the video playback thus simulating a change in vehicle speed. At a road junction the user can choose the next direction to take. In doing so they determine the next video footage to be played back. Lippman's work also allows users to stop journeys and inspect interesting buildings passed which introduces other media (still pictures and text) into the experience. Unlike Sawhney's café scenario, Lippman's video-map space is organised as a grid of straight roads which means that directional decisions at junctions of roads are only ever forward, backward, left (90° anti-clockwise), or right (90° clockwise). Sawhney's navigation is based around selecting the next table in the café to walk near to in order to eavesdrop on the conversation taking place at that table.

Sawhney's hypervideo framework [173][174] used to construct their café scenario supports spatial link opportunities and the grouping of segments into spatial areas. This could be used to support the construction of video-maps discussed by Lippman. Essentially, Sawhney's work subsumes Lippman's as it provides a more general framework for the construction of video-maps.

6.3.3 Video-Map Scheme and Schema

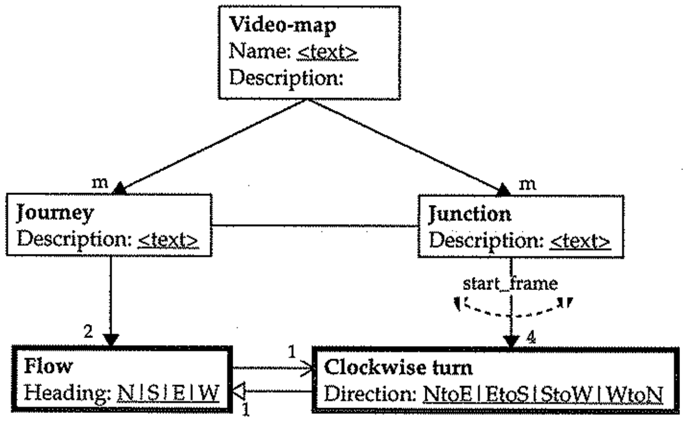
As stated at the start of this chapter, the thrust of this thesis is not the extensive analysis and capture of video material itself, rather how such material can be structured and reused. To this end, this example is based upon the work of Lippman as it provides a simpler spatial arrangement (a grid structure as opposed to a web like structure). Also, less work is involved in capturing the required video as only journeys and junctions need to be recorded, instead of additionally recording the many possible conversations of Sawhney's café. To conclude, this example scheme has certain interesting properties for the thesis and involves less video capture and analysis than Sawhney's video-maps.

Figure 46 illustrates a VCMF schema representing the structure of video-maps (the video-map schema) which is detailed in Appendix E (pp. 310). Here a video-map is considered to be made up of Journeys (straight paths in this example) which meet Junctions (the meeting is expressed using the Meets relationship) where the user may choose their next direction. These Meets relationships are represented by Schema specific relationships which are one of the important structural properties exhibited by this example. Each Journey is made up of two Flow Units which have an attribute for the heading of the video footage (North, South, East or West); there is one Unit for each direction along the path. Each Junction is made up of a sequence of 4 Turn Units which represent the 4 clockwise 90° turns required to turn through 360°. These Turn Units have a direction attribute which describes the turn being made in terms of compass directions (*e.g.* NtoE means a clockwise turn from facing North to facing East). Each Flow Unit has an Entry relationship to a Turn Unit which the user would see as their first clockwise turn at a junction. Likewise, an Exit relationship exists between a Turn Unit and the Flow Unit which the user would see as they exited the junction from that turn. These Exit and Entry relationship types are Schema specific relationships; they describe relationships which are not Membership, Sequential, or Temporal.

Note that this thesis is solely concerned with the use, and modelling of, video and so this schema does not provide any support for Lippman's multimedia information (text and graphics) about interesting points within video-maps.

6.3.4 Video-Map Model and Instance

Unlike Lippman's car centred video-map, this example is pedestrian based. This makes recording the video for the paths and junctions much easier as no vehicles are involved. A small tree lined square provides the space for this example video-map. It is



Relationship types	a ——▶ b	a - - - -▸ b	a —— b	a ——▷ b	a ——> b
Name	*a* Contains *b*	*a* Leads_to *b*	*a* Meets *b*	*a* Enters *b*	*a* Exits_to *b*
Inverse name	*b* is_a_member_of *a*		*b* meets *a*		

Figure 46: VCMF video-map scheme

enclosed (so providing a finite map), and the paths around the square are organised in a grid like structure which suits Lippman's approach. Figure 47 is a map of the space used in this example (Queen's Square, located in central London). There are two main paths in this square: a path that runs centrally from the East gate to the West gate, and a path which runs around the perimeter of the square. A notable feature is the rockery in the north east corner of the square which Lippman's system may have stored multimedia data about. The dotted lines at the West gate indicate that the exit is not modelled because of difficulties recording the footage at that point.

The diagram in Appendix D (pp. 283-284, 297-298) shows the instance representing the video-map of Queen's Square (the video-map instance) which was constructed using the VCMF toolset. The video footage was recorded using a Hitachi Video Camera/ Recorder Model: VM-s7200E(UK) which produced a total of approximately 8 minutes footage for the whole square. This footage was captured by placing the camera at shoulder height and walking up and down each path, and rotating through 360° at each corner, including the two gates. The problems of video-map footage capture discussed by Lippman [121] were evident in the capture of footage for this example. In particular, even within the short period of time it took to capture the footage (around 50 minutes), the weather conditions changed which caused noticeable differences in lighting between parts of the video-map. Other problems included the number of people in the square as the footage was captured. Even though there were typically less than five people in the square at any one time they moved around which causes interesting continuity problems when viewing the video-map; one person may appear to be in several places at once. However, none of these problems affect the video-map instance as it is only concerned

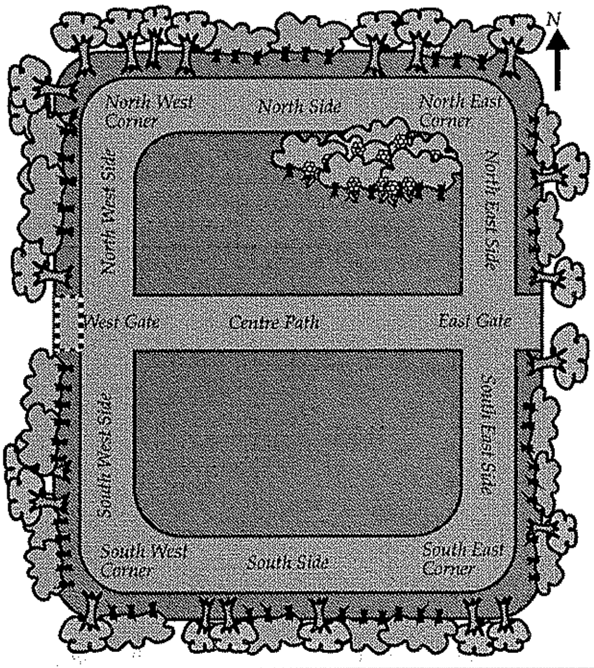
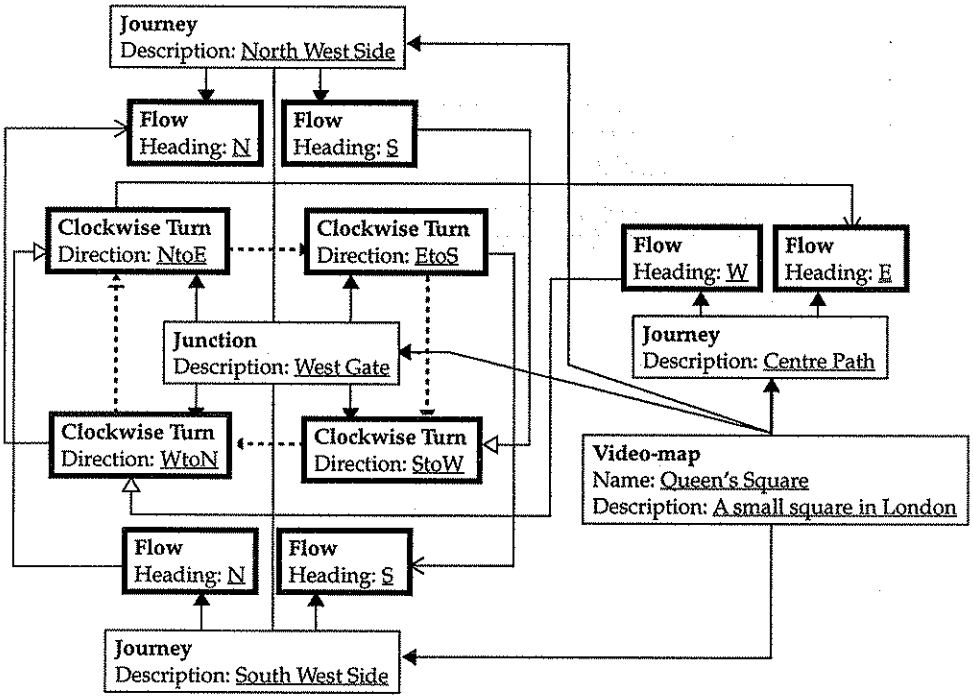


Figure 47: Map of Queen's Square

with the spatial location of footage rather than the weather conditions, people present, or lighting of the environment. For this reason, it was decided to use the visually inconsistent footage rather than attempting to create a seamless experience which Lippman had to go to great lengths to achieve. This footage was digitised and stored using the same technique as the previous example (frame rate: 10 frames per second, frame size: 192 by 144 pixels, total size: 13.2 Mb). In total the model contains 62 objects and 130 relationships.

A special navigation interface was constructed to provide easier navigation within the video-map instance (discussed in Chapter 5). This interface presents the user with information about their current position and heading, and allows them to select their next direction at the appropriate points. Of course, the Simple Navigation component discussed in Chapter 5 could be used to navigate around the video-map, but the specialised interface is easier to use as it is tailored to the task of navigating through video-maps. The ability to have different user interfaces for the same instance highlights the strength of VCMF and the implementation approach.

Figure 48 is an extract from the video-map instance presented in Appendix D. The figure shows the West Gate Junction and the objects it is related to (all other relationships and objects are not shown to simplify the diagram). A user at the West Gate, facing North, could chose to turn clockwise (NtoE), or to travel North along the North West Side. This is non-deterministic presentation as it is not possible to determine beforehand which choice the user will make.



Relationship types	a ——▶ b	a - - - - -▸ b	a ——— b	a ——▷ b	a ——> b
Name	*a* Contains *b*	*a* Leads_to *b*	*a* Meets *b*	*a* Enters *b*	*a* Exits_to *b*
Inverse name	*b* is_a_member_of *a*		*b* meets *a*		

Figure 48: Extract from video-map instance

6.3.5 Reuse of the Video-Map

Two examples of reuse are considered in this sub-section. The first involves the user of the video-map instance creating *guided tours* for later use. The second deals with the creation of a documentary about the area captured in the video-map.

6.3.5a Reuse in a guided tour of the video-map

Hypermedia maps often allow users to create or use *guided tours* through the spaces represented. A guided tour provides a pre-determined linear journey through a space; there is no need for the user to make decisions about which direction to follow through the tour. For example, the interactive map of Amsterdam [161] supports virtual tram journeys through Amsterdam. These are linear sequences of parts of a spatially arranged map of Amsterdam. Likewise, it would be useful to support the creation of guided tours through the video-map.

Figure 49 illustrates a VCMF schema to represent guided tours of video-maps (the guided tour schema). The Guided Tour Collection is a User-Ordered Collection as the user determines the sequence of Flow and Turn Units within it.

Appendix D (pp. 283-284,297-298) illustrates a guided tour of the video-map instance. For simplicity, Figure 50 shows this guided tour as an extract; the objects and

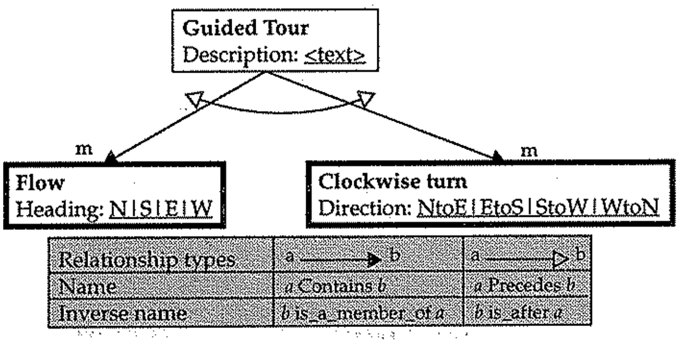
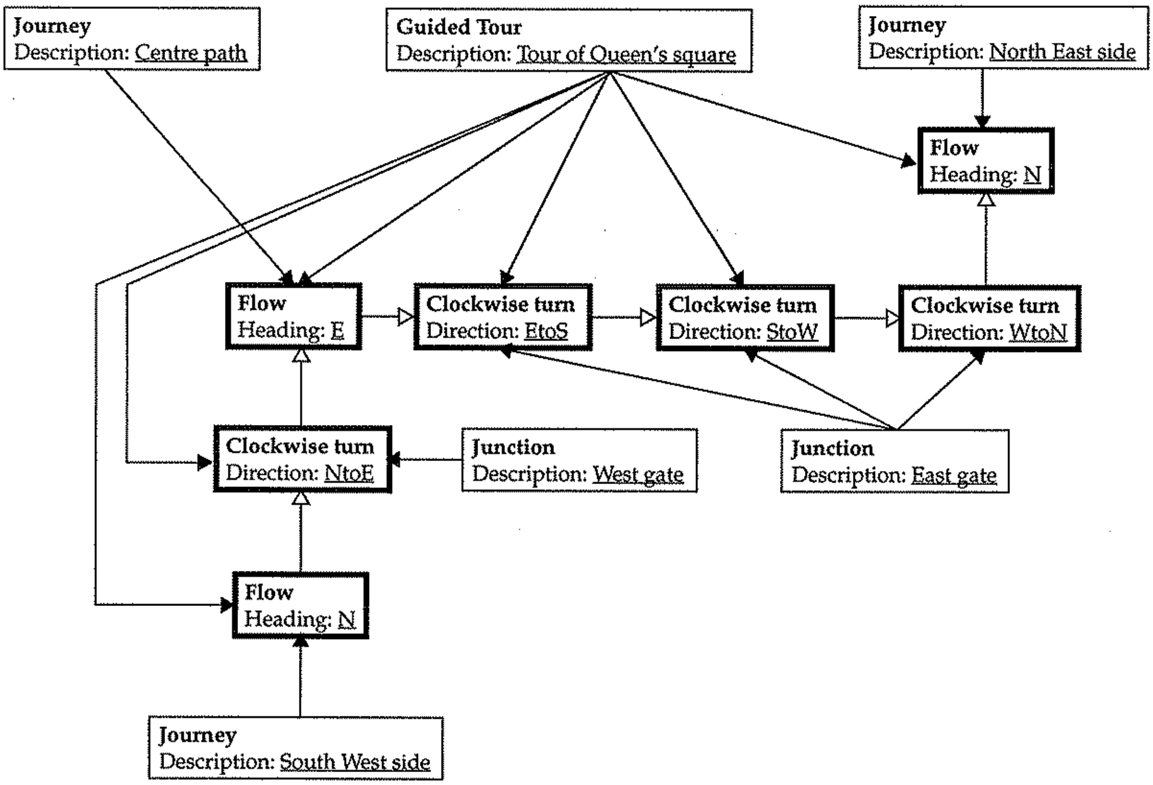


Figure 49: Guided tour schema

relationships involved in the guided tour are shown as well as the Collections that the Flow and Turn Units are members of so that their position in the video-map is illustrated. The tour starts at the South end of South East side (at the start of the North Flow Unit). It travels via the Centre Path (along East Flow), and ends at the North end of the North East side (at the end of the North Flow). The Turn Units needed to move from one Flow Unit to another are shown in the diagram. There are 3 Turn Units used at the East gate as they turn in a clockwise direction and the heading needs to change from Eastwards to Northwards. Currently this guided tour must be manually constructed by the user, but the video-map navigation interface could easily be modified to include objects viewed by the user in the guided tour. Viewing the guided tour instance simply involves the user requesting the playback of the Guided Tour Collection's member objects by following the Precedes relationships. The Simple Navigation component discussed in Chapter 5 can be used to support such viewing.

6.3.5b Reuse in a documentary about London

The second example of reuse involves the construction of a small documentary discussing the interesting flora to be found around London. This documentary includes footage from the video-map which shows the rockery in the North East corner of Queen's Square (shown at the top right hand corner of Figure 47). The schema used to construct the previous documentary (the modified plot segmentation schema illustrated in Figure 43) must be modified to support the construction of a documentary which includes objects from the video-map. Figure 51 illustrates this modified schema. Two object types (Flow and Turn Units) from the video-map schema have been added as members of the Sequence and Scene Collections. The need to modify the plot segmentation schema illustrates one of the main issues raised when reusing schemas in different applications and is discussed in Chapter 7. Although this reuse of the plot segmentation schema requires modification of the schema, it is used as an example of reuse as it shows the use of VCMF to support an important kind of reuse: the reuse of objects from one instance in a different application, and in a different schema. As mentioned previously, and



Relationship types	a ──▶ b	a ──▷ b
Name	*a* Contains *b*	*a* Precedes *b*
Inverse name	*b* is_a_member_of *a*	*b* is_after *a*

Figure 50: Guided tour extract from video-map instance

discussed in Chapter 7, this important kind of reuse is not supported by other video content modelling systems.

The instance representing this documentary is illustrated in Appendix D (pp. 299); an extract this instance is shown in Figure 52. Table 17 details the segmentation of the documentary in terms of Parts, Sub-Parts and Scenes. The documentary itself has a categorical structure and was created specifically to show the reuse of objects from the video-map instance in an instance of a different schema, and in a different application. Although the documentary is small in size compared to the documentary in the previous section, it is not unrealistically small and contrived. Many documentaries broadcast on television are short and cover a small subject matter. For instance, *The Weather Show* discusses a topic connected to the weather, lasts for five minutes, and is currently shown on weekday lunchtimes on BBC 1.

The extract in Figure 52 shows how a Turn Unit from the video-map instance (turning from South to West) is reused in the documentary to illustrate some of the flowers in one of London's squares (Queen's square). In keeping with the previous example of reuse in a documentary (illustrated in Figure 44) the thick lines in the diagram indicate which

759(b)

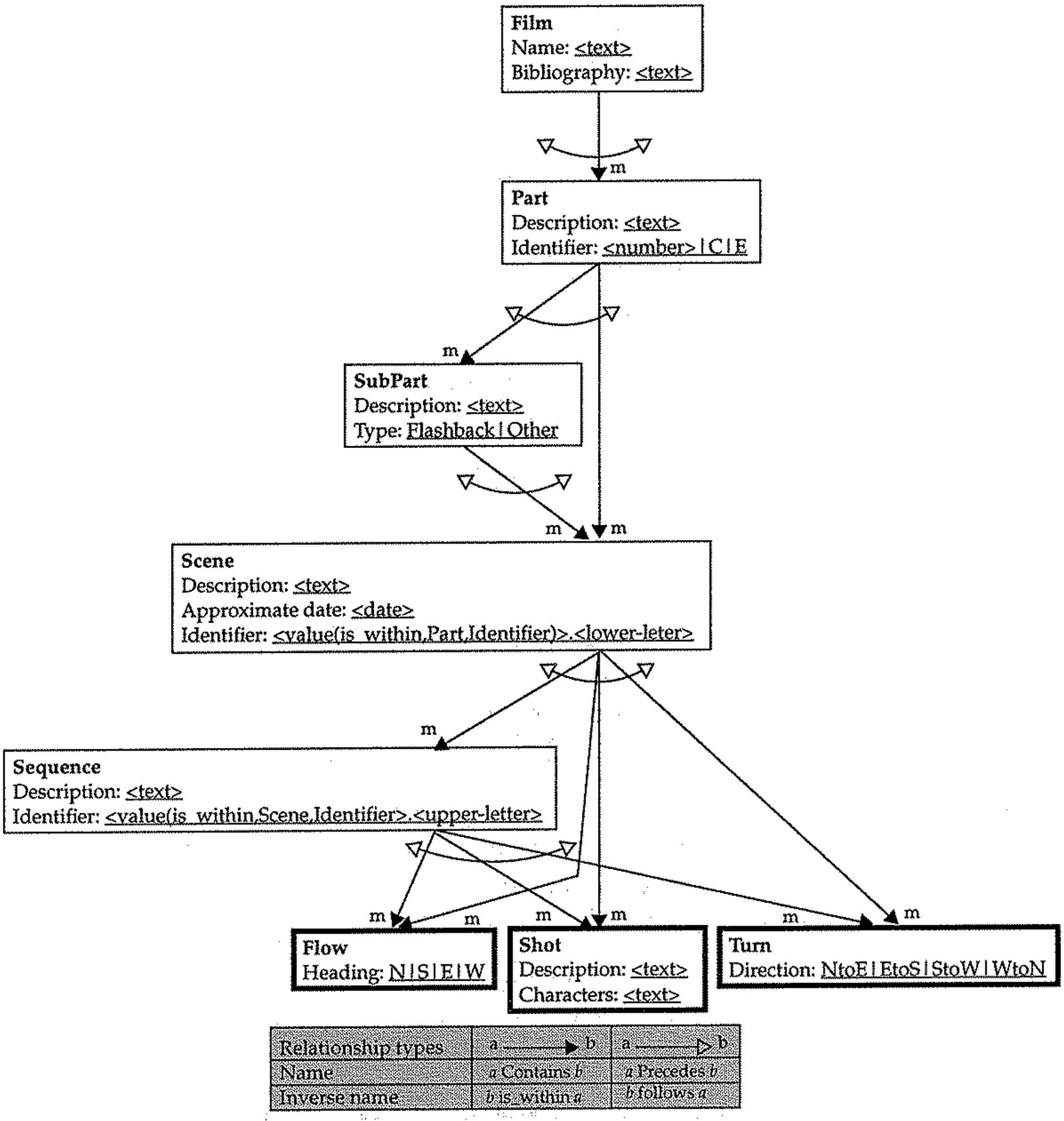


Figure 51: Modified plot segmentation schema to support documentary of video-map

relationships facilitate the reuse of objects. In this case a Shot of the narrator discussing ivy is followed by a Turn Unit which shows some ivy in Queen's Square.

As with the previous example of reuse in a documentary, this reuse shows objects of one instance (the video-map) being mixed with objects from another instance (the documentary). In contrast to the previous documentary example, this example shows how objects created within one schema can be mixed with objects created from a markedly different schema (the previous example used a slightly modified schema). Investigating VCMF's ability to support this kind of reuse is important to this thesis as it is poorly supported by other approaches.

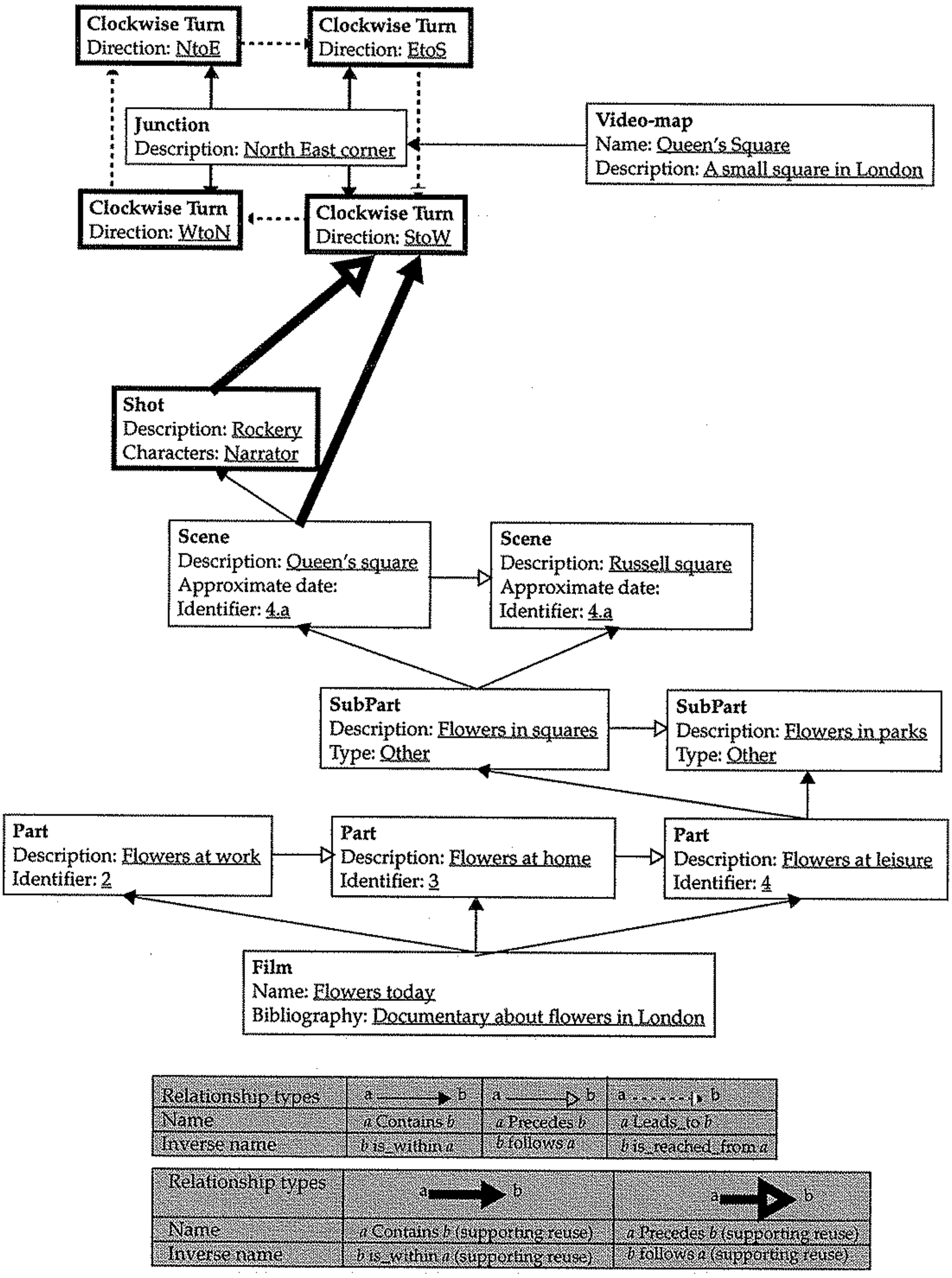


Figure 52: Reuse of video-map objects in a documentary

6.3.6 Summary: Properties Exhibited in the Video-Map Example

As with the previous example, this example shows VCMF's ability to model both homogeneous and heterogeneous collections, and 1-m membership relationships. Here the similarities end. This example shows VCMF's ability to support non-deterministic presentation, and to describe the following structural properties not present in the first

Part	Sub-Part	Scene
C. Credit title		
1. Introduction		
2. Flowers at work		a. Flowers at QMW
3. Flowers at home		a. Houseplants
4. Flowers ay leisure		
	Flowers in Squares	a. Queen's Square
		b. Russell Square
	Flowers in Parks	c. Regent's park
		d. Hyde park
5. Summary		
E. End credits		

Table 17: Plot segmentation of documentary about flowers in London

example:

1. **1-x membership relationships**

   The Journey and Turn Collections are related to their member objects by relationships with 1-2 and 1-4 cardinality respectively.

2. **Unordered member objects**

   The Video-Map Collection does not order its member objects in any way.

3. **Arbitrary ordering of member objects**

   The Guided Tour Collection's member objects are arbitrarily ordered by the user at run time.

4. **Schema specific relationships**

   Several Schema specific relationships types (Meets, Entry, Exit) are introduced in the schema.

This example shows the following kinds of reuse:

1. **Reuse of objects within the same application (different schema)**

   The first reuse in this example shows VCMF's ability to support a user's arbitrary reuse of objects in a different schema (the guided tour schema) within the same application. It is similar in nature to the reuse exhibited in the viewing of Kane's life in chronological order of flashbacks in the previous example.

2. **Reuse of objects in a different application with a different schema**

   This second example of reuse is very important as it shows that VCMF can support reuse of objects by mixing objects of different instances, and different schemas. In this form of reuse objects from the video-map instance are reused to help create a documentary. This documentary is a different application and uses a different schema.

In summary, this example demonstrates further properties of video content models that VCMF can describe. This is the middle stage of demonstrating VCMF's ability to describe a cross section of schemes. Even at this stage VCMF's utility is evident as it has been able to describe two markedly different schemes which previously would not have been possible using one approach. The next section describes an example of VCMF's use which exhibits properties including those not exhibited in these first two examples.

6.4 Style Analysis

6.4.1 Purpose of the Style Analysis Example

This example demonstrates VCMF's ability to describe pre-defined sequences of objects within collections and Temporal relationships between member objects. As such it completes the cross section of characteristics discussed in Section 6.1 and completes the support of the thesis. This example also shows a form of reuse not shown before: the reuse of video footage (not including any objects) in a different application using a different schema. Furthermore, it shows a second example of the most important form of reuse: the reuse of objects from one instance in another instance created from a different schema, and used for a different application.

6.4.2 Domain of Style Analysis

The scheme in this example is designed to support the analysis of style in narrative films.

Analysis of a film's style differs from the analysis of its plot as the analyst is not interested in the use of Scenes and Shots *etc.* to advance the plot. Instead they are interested in drawing out the patterned and significant use of techniques within the film and comparing and contrasting these techniques in order to examine their functions within the film. Techniques examined with respect to style are: *Mise-en-scene* (the settings, props, lighting, costumes, make-up *etc.*), *cinematography* (the way in which the camera moved whilst recording, and the ways in which the film was manipulated during development), *editing* (the construction of sequences of Shots), and *sound* (the use of sound recordings synchronised with the film for playback).

These techniques do not necessarily fit into the general structure of plot segmentations as many different stylistic techniques (*e.g.* many different cinematography techniques) may be used within one Shot (regarded as an atomic unit in a plot segmentation), or one stylistic technique may cover several Shots without a break but may change before the next Sequence/ Scene (*e.g.* one musical technique may span several Shots but may change within a Scene). This difference is reflected by the contrasting schemes used in stylistic modelling systems such as Media Streams [55] and plot based modelling systems such as the Video Database Browser [165]. Moreover, unlike a scheme for plot analysis, a scheme to support analysis of a film's style must be able to describe patterns of use of salient techniques. For these reasons, a different scheme is needed when analysing a film's style as opposed to analysing a film's plot.

Bordwell *et al.* [21] propose four steps in the analysis of a film's style:

1. **Determine the organisational structure of the film**

The first step (in the case of analysing a narrative film) is to conduct a plot analysis of the film in order to understand how the film is put together as a whole.

2. **Identify the salient techniques used**

Here the analyst looks for the salient techniques used, either to show that the film fits into a particular stylistic group, or that it stands out from other films in its use of stylistic techniques.

3. **Trace out patterns of techniques within the whole film**

The patterns of salient techniques identified in the previous step are identified and drawn out throughout the whole film.

4. **Propose functions for the salient techniques and the patterns they form**

The purposes of techniques and patterns of techniques throughout the whole film are proposed; examples of them throughout the film are located and used to support these proposals. Parallels between patterns are particularly important here, as is the comparison between patterns and their position within the organisation of the film (the plot segmentation).

Style analysis is similar to plot analysis in terms of its domain of use (typically the academic analysis of films), but the characteristics of schemes used are significantly different. A completely different domain such as the sociological analysis of behaviour captured in video recordings (discussed in Chapter 2) could have been used instead of style analysis to exhibit a similar set of structural attributes. The schemes used in such a domain must be able to describe patterns of annotations of video, and describe the rich temporal relationships between annotations [88]. The domain includes systems such as Harrison's VANNA system [88] which supports video annotation aimed at supporting sociological data analysis; Fisher [62] provides a general discussion about the iterative, *ad-hoc* nature of such analyses. However, style analysis was chosen over such a domain for the following reasons:

1. Unlike sociological data analysis, there is plenty of data (descriptions and source video material) readily available concerning style analyses of films. This means that this thesis concentrates on how VCMF can support the attributes and reuse exhibited, rather than on conducting a sociological data analysis.
2. A style analysis could be conducted on *Citizen Kane* in which case both the plot segmentation, and the video itself could be reused thus reducing the amount of storage space required to conduct the demonstration. Moreover, this reuse supports the investigation of interesting kinds of reuse not shown in the other examples.

6.4.3 Style Analysis Scheme and Schema

A scheme to support the style analysis of films contains three interrelated parts to support the four stages described in the previous section. The first stage requires the scheme discussed in Section 6.2 to support determining the organisation of the film by conducting a plot segmentation.

The second stage requires support for identifying and noting salient techniques used in the film. A scheme is required which defines the different cinematic techniques and their categorisation and whose models are able to describe techniques used in a particular film. The scheme inbuilt in Davis' Media Streams [55] and Davenport *et al.*'s discussion of cinematic strata [51] would support these descriptions. These two schemes are discussed in detail in Chapter 2; their applicability to this particular use is discussed here. Both schemes categorise annotations of the content of a Shot into objects (characters or props) and their position and actions, cinematographic techniques used, recording medium used, and mise-en-scene. Media Streams places more emphasis on the transitions between Shots (the editing) than cinematic strata, but has no representation of sound. Together they represent all the cinematic techniques discussed by Bordwell *et al.* [21] with respect to the style analysis of films. Table 18 summarises the four categories of techniques discussed by Bordwell *et al.* and Davenport *et al.*, and implemented by Davis. Media Streams concentrates on changes in techniques, for example, zooming in on an object, which makes it a particularly suitable system to support this part of style analysis as stylistic effects are typically caused by changes in techniques rather than the techniques themselves.

The third and fourth stages require a scheme which can support the description of patterns of salient techniques identified in a film, and the effects they create. These effects are often not simply a result of patterns of one technique (such as the repeated use of one kind of editing), but may be created from the combination of patterns of several techniques (such as the use of editing and cinematography). This makes describing patterns of techniques difficult as different combinations of techniques may be used to create the same effect. For instance, the effect of moving into a space can be achieved using either cinematographic techniques such as trucking forward or zooming in, or editing closer Shots together, or a combination of these two categories of techniques. The scheme essentially needs to describe constituent parts of an effect, and how they are patterned to cause the effect. In the model, these effects need to be related to the techniques used to create them. For instance, moving into a space typically starts from one position, moves through the space using a combination of cinematic and/or editing techniques, and terminates at another position. In this example, the scheme needs to describe the fact that there are three stages, a beginning, middle, and end, and that the effect produced is that of movement into a space caused by a combination of cinematic and editing techniques.

An example of a stylistic pattern introduced by Bordwell *et al.* [21] is the repeated 'penetration into the space of a scene' using cinematographic and editing techniques in *Citizen Kane.* Each penetrative pattern starts with the camera focused on a stationary

Category of techniques	Techniques
Cinematography	Lens focal length - zoom in/out
	Truck - change position or height
	Tripod - pan or tilt
	Framing
Editing	Cut
	Dissolve
	Fade in/out
	Wipe
Mis-en-scene	Setting
	Costume and make-up
	Lighting
	Figure expression and movement
	Space
	Time
Sound	Acoustic properties - loudness, pitch, timbre
	Rhythm
	Fidelity
	Space
	Time - synchronous or asynchronous sound

Table 18: Categories of stylistic techniques used in film making

object, the camera then moves forwards (possibly using several Shots) until finally arriving and focusing on the main item of interest in the Scene. This ingress pattern, and its complement withdrawal through a space, are used as the main examples of style analysis in this section.

6.4.3a Style analysis schema

Figure 53 illustrates a VCMF schema used to support style analysis (the style analysis schema) which is detailed in Appendix E (pp. 311-313, 315-317). It is the combination of the three parts discussed in this section; plot segmentation (for simplicity, only one object type of the plot segmentation schema is shown, the others are assumed to be present), description of techniques used, and descriptions of patterns used to create particular effects. The schema concentrates on the analysis of cinematographic and editing techniques, and the identification of patterns of ingress and withdrawal. The figure shows how the schema provides categorisation of the techniques using the Un-Ordered

Collections Cinematography, Editing, Sound, and Mis-en-scene. All the techniques of the first two categories have been represented. The others have been left empty for brevity as they are not used in the two example patterns. A Schema specific relationship type Uses is illustrated in the schema to relate the patterns of effects to the categories of techniques they use.

The two patterns of ingress and withdrawal are represented by the Ingress and Withdrawal Collections shown in the diagram. These are both composed of a sequence of Units as follows:

- First one From Unit which represents the first object the camera focuses on
- Second, one or more Via Units which represent the objects passed on the journey to the final object.
- Finally, one To Unit which represents the object the camera finally focuses on in the sequence.

This sequence of objects is reflected in the instance; the schema constrains the possible sequence of objects to satisfy the pattern being analysed.

6.4.4 Style Analysis Model and Instance

Appendix D (pp. 285-291, 299-305) details the instance used to support the style analysis of *Citizen Kane* (style analysis instance). Conducting a style analysis of *Citizen Kane* means that the digitised recording used for the plot segmentation can be reused, the plot segmentation can be reused, and there is plenty of literature describing the interesting stylistic patterns used. The style analysis instance contains 217 objects and 341 relationships.

This instance supports the four stages discussed in Section 6.4.2 as follows:

1. Determine the organisational structure of the film

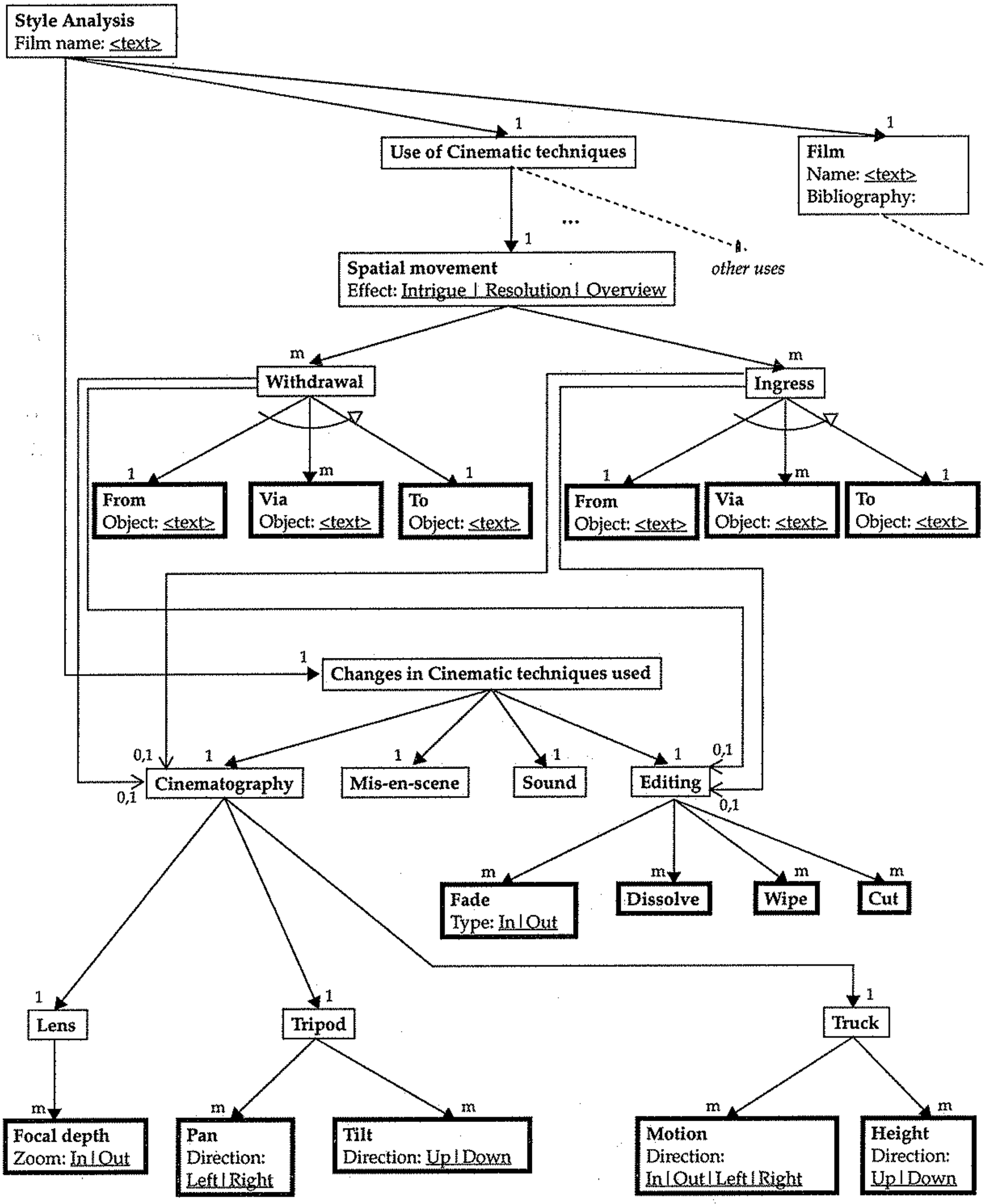
The instance includes the plot segmentation conducted in Section 6.2 in the Un-Ordered Collection Style Analysis. This reduces the amount of effort involved in conducting the style analysis. This inclusion of the plot segmentation instance supports investigation of patterns and their position within the organisation of the film discussed in stage four.

2. Identify the salient techniques used

The instance shows how the categories of techniques and individual techniques used in the film are represented. Identifying and annotating all the techniques used is a time consuming and tedious operation, but systems such as those discussed by Tonomura *et al.* [195] could be used to automatically generate this part of the instance. This would considerably reduce the analytic effort of the analyst who would then mark which techniques are considered salient.

3. Trace out patterns of techniques within the whole film

The patterns of ingress and withdrawal discussed previously are described in the instance, and related to the techniques that are used to create these effects.



Relationship types	a ──▶ b	a ──▷ b	a ──> b
Name	*a* Contains *b*	*a* Precedes *b*	*a* Uses *b*
Inverse name	*b* is_within *a*	*b* follows *a*	*b* is_used_by *a*

Figure 53: Style analysis schema

For example, Figure 54 illustrates one description of ingress in *Citizen Kane*. In this example the viewer gets their first view of Susan (Kane's second wife who became a drunken night club singer). The ingress starts with a view of a billboard showing Susan's face (shown in close-up in Figure 55a and described by the From Unit of the instance). The camera then trucks up and in[1], passing over the billboard (the top of the billboard is shown in Figure 55b, the movement is described by the first Via Unit of the instance). Next it tilts down and trucks down through the nightclub sign and the skylight of the nightclub (the sign is shown in Figure 55c, the skylight is just visible at the bottom of this figure, this movement is described by the second Via Unit). A Dissolve is then used to reveal Susan lying slumped on a table (seen in the bottom left of Figure 55d and described by the To Unit of the instance).

The Spatial Movement Collection (First view of Susan) is related to Cinematography and Editing Collections by Uses Schema specific relationships. These Collections detail the salient techniques used to create the effect: truck motion and height changes, tripod tilt, and an edit dissolve. As mentioned previously, Figure 54 does not show the overlapping nature of these techniques. Figure 56 illustrates the overlapping nature of these techniques using a timeline. Timelines are discussed in Chapter 2, and are used in systems such as Media Streams [55]. Moreover, the VCMF toolset discussed in Chapter 5 includes a component which can create timelines of instances.

Essentially, in a timeline, the objects are represented on a temporal axis; their temporal extent is represented by a rectangle. To simplify this figure the object's type name is given along with the key unique attribute value (if any) from the instance. The figure shows the initial From Unit which is followed by an increase in truck height as the viewer is moved Via the top of the billboard. The viewer is then drawn Via the sign and the skylight using a truck in overlapped at the end by a tilt down of the tripod and a decrease in the truck's height. Finally a dissolve is used to reveal the destination of the Ingress, Susan slumped on a table of the night club.

The timeline representation is interesting as it allows the analyst to view the temporal overlap between techniques. However, in this situation it is not the temporal relationships between the techniques that are of interest, but the patterns of Ingress and Withdrawal, and the techniques they use. The timeline in this situation provides additional information about how these effects are created.

4. Propose functions for the salient techniques and the patterns they form

Bordwell *et al.* propose that the patterns of ingress into the space of a Scene function to create curiosity and suspense; the camera moves towards things that might reveal secrets of Kane's character. Similarly, an analyst might suggest that the patterns of withdrawal are used to resolve questions about things in Scenes. In contrasting these two

[1] Note that the sequence of techniques used is not illustrated in Figure 54. The start_frames and end_frames of the technique objects are fully illustrated in Appendix D.

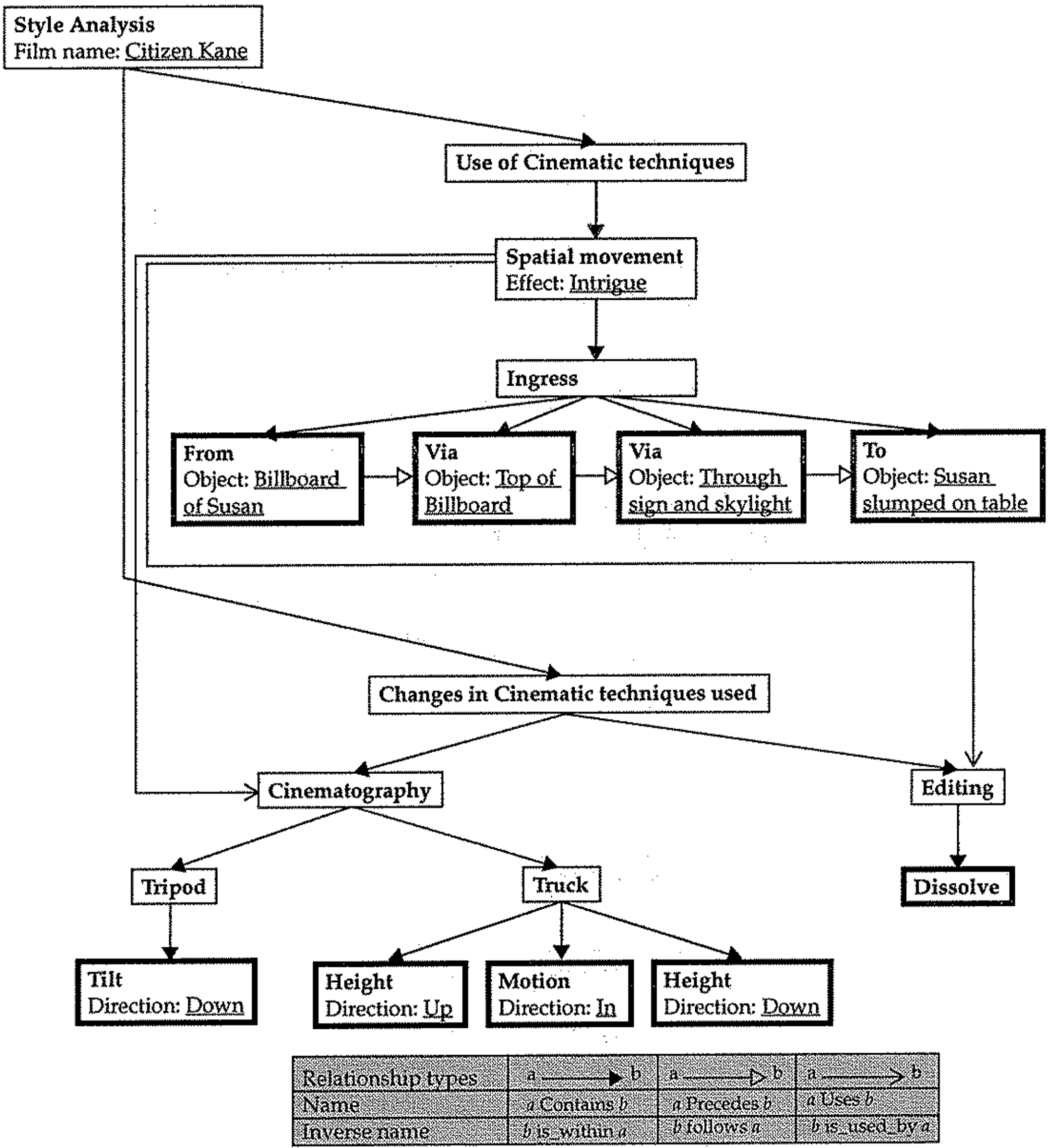
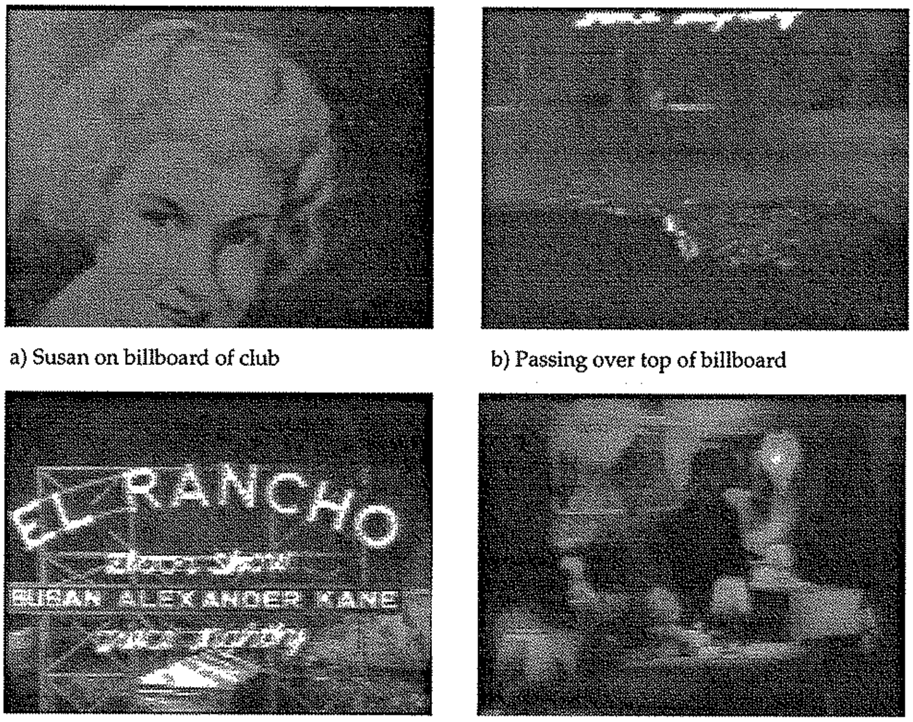


Figure 54: Extract from *Citizen Kane* style analysis model: an example of ingress

patterns an analyst might speculate that ingress occurs at the beginning of Scenes in order to intrigue the viewer about what is about to happen in the Scene, whereas withdrawal occurs at the end of Scenes in order to resolve questions that might have been raised in the Scene, or previous Scenes.

The first step in investigating this proposition is to determine if it is true that ingress usually happens at the start of Scenes, and withdrawal usually happens at the end of Scenes. This form of query is supported by the Temporal Collections of VCMF. Figure 57 shows the schema used to support this investigation. For simplicity the other object types are not included in this figure, but are assumed to exist. When the Comparisons Collection is instantiated and all Scene, Ingress, and Withdrawal objects are added, the



a) Susan on billboard of club

b) Passing over top of billboard

c) Nightclub sign and skylight

d) Susan slumped at table

Figure 55: Stills from the first view of Susan in *Citizen Kane*

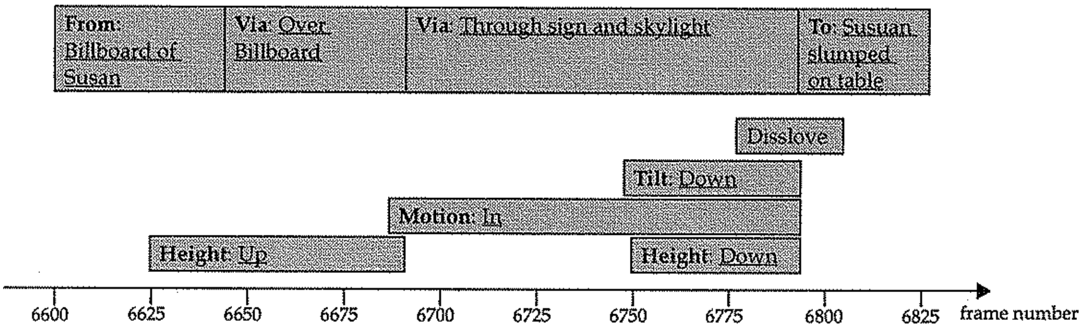


Figure 56: Timeline showing an example of ingress and the techniques used to create it

VCMF system instantiates all the appropriate temporal relationships between members of the Collection. The analyst then examines these relationships to determine if the first part of the proposition is true. Table 19 summarises the patterns of withdrawal in *Citizen Kane*, similarly, Table 20 summarises the patterns of ingress (all detailed in Appendix D). The table shows the pattern's name, the Scene(s) it is in (described by the Scene number followed by its description), the temporal relationship that exists between the pattern and the Scene, and the effect the analyst considered the pattern to have. From these summaries no useful conclusions about the proposition can be drawn as 4 out of 6 withdrawals appear at the end of Scenes, and 6 out of 9 ingresses appear at the start of Scenes. The important aspect of this investigation is not what conclusions are drawn from the analysis, but rather the fact that VCMF supports the analysis.

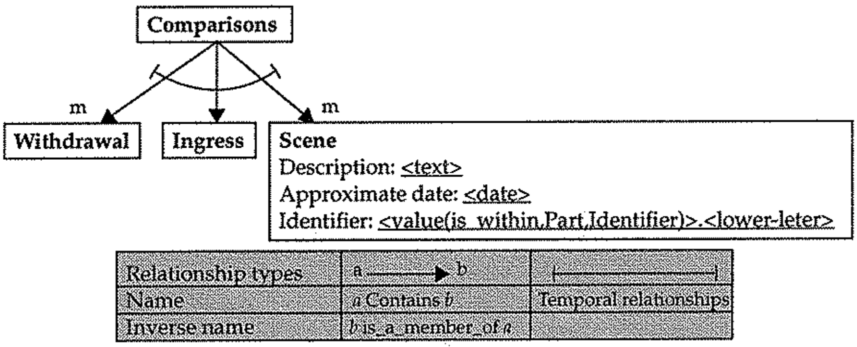


Figure 57: Schema to support investigation of purpose of ingress and withdrawal

Pattern	Scene	Temporal relationship to Scene	Effect
Start of new Inquirer	5.c: Montage: The Inquirer's growth	Starts	Overview
Realisation of Kane's marriage problems	6.b: Breakfast table montage: Kane's marriage deteriorates	Ends	Resolution
Kane's affair discovered	6.f: Kane confronts Gettys, Emily, Susan 6.g: Kane loses election and Leland asks to be transferred	Overlaps	Resolution
Thompson leaves Susan	7.l: Susan concludes her reminiscence	Ends	Overview
Final overview of Kane's possessions	8.c: Raymond concludes his reminiscence, Thompson talks with the reporters, all leave	Ends	Overview
Last view of Xanadu	8.d: Survey of Kane's possessions leads to a revelation of Rosebud. Exterior of gate and castle. The end	Ends	Resolution

Table 19: Patterns of withdrawal in the film *Citizen Kane*

Investigating the second part of the proposition involves the analyst's judgement. Each Ingress and Withdrawal Collection needs to be viewed, and its effect determined. In particular, the analyst needs to question whether a particular ingress provokes intrigue about something in the Scene, and conversely, whether a particular withdrawal resolves some question raised earlier in the film. In *Citizen Kane* it is the case that ingress provokes intrigue about something in the Scene and withdrawal often resolves queries about something previously presented in the film. As illustrated in Table 19, it is also the case that withdrawals often give overviews of the position of characters in a Scene.

Pattern	Scene	Temporal relationship to Scene	Effect
First view of Kane - on death bed	1.a: Xanadu: Kane dies	Starts	Intrigue
First view of Susan	3.a: El Rancho nightclub: Thompson tries to interview Susan	Starts	Intrigue
First flashback about Kane (as a boy)	4.a: Thompson enters and reads Thatcher's manuscript 4.b: Kane's mother send the boy off with Thatcher	Overlaps	Intrigue
Kane's first edition completed	5.b: Kane takes over Inquirer	Is_during	Intrigue
Kane meets Susan	6.d: Kane meets Susan and goes to her room	Starts	Intrigue
Thompson's second visit to Susan	7.a: Thompson talks with Susan	Starts	Intrigue
Susan bored in Xanadu	7.g: Xanadu: Susan bored	Starts	Intrigue
Kane and Susan have a picnic	7.j: Picnic: Kane slaps Susan	Is_during	Intrigue
Burning of Rosebud sledge	8.d: Survey of Kane's possessions leads to a revelation of Rosebud. Exterior of gate and castle. The end	Starts	Intrigue

Table 20: Patterns of ingress in *Citizen Kane*

Systems such as TVQL [91] could be used to support the investigation of the first part of the proposition. As discussed in Chapter 2, such systems allow 'users to pose queries, as well as to *browse* the data in a *temporally continuous* manner, thereby aiding them in the discovery of temporal trends' [91]. The Temporal Collection supports this discovery of temporal trends through browsing of the Temporal relationships generated.

6.4.5 Reuse of the Style Analysis

Unlike the other two examples, this example's reuse is intertwined with the instance itself. There are three kinds of reuse shown in this example. First, the video of *Citizen Kane* is used as the basis of both the plot and style analyses. Second, Scenes from the plot segmentation instance are reused in the style analysis instance when considering the purpose of stylistic patterns (ingress and withdrawal) in the film. This form of reuse is the reuse of objects from one application in another, and with a different schema. Third, the consideration of these purposes involves the reuse of objects within the instance being reused in another instance (the Comparisons Temporal Collection). In contrast to the previous example of reuse, this reuse involves the reuse of objects in the same application, but different schema.

6.4.6 Summary: Properties Exhibited in the Style Analysis Example

This final example is used for analysis, and supports the investigation of temporal relationships between objects. It shows the following structural properties:

1. **Homogeneous collections**

The Lens object type has homogeneous members.

2. **Heterogeneous collections**

Apart from the Lens object type, all Collections in this schema have heterogeneous members.

3. **1-m membership relationships**

The Collections Spatial Movement, Withdrawal, Ingress, Editing, Lens, Tripod, and Truck are related to at least one of their member objects using relationships with cardinality 1-m.

4. **1-x membership relationships**

The Collections Changes in Cinematic Techniques Used, Uses of Techniques, Withdrawal, Ingress, Style Analysis, and Cinematography are related to at least one of their member objects using relationship with cardinality 1-1. Additionally, the Uses Schema specific relationship has cardinality 1-1 or 1-0 indicating that it need not be instantiated in the instance.

5. **No sequence of member objects**

The object types Style Analysis, Uses of Techniques, Spatial Movement, Changes in Cinematic Techniques Used, Cinematography, Editing, Lens, Tripod, and Truck do not impose any ordering on their member objects. This is used in Collections such as Cinematography to represent the categorisation of cinematographic techniques.

6. **Pre-defined sequences of member objects**

Object types Withdrawal and Ingress both pre-define the sequence of their member objects when instantiated. This is used to draw out the patterns of effects exhibited in the film being analysed.

7. **Temporal relationships between member objects**

The Comparisons Temporal Collection is used to investigate the proposition that ingress occurs at the start of Scenes whereas withdrawal occurs at the end of Scenes. It relates it member objects (Ingress, Withdrawal, Scene) to each other using Temporal relationships.

8. **Schema specific relationships**

A single Schema specific relationship Uses is introduced in the schema.

The reuse in this example shows VCMF's ability to support the following kinds of reuse:

1. **Reuse of video footage in a different context using a different schema, and for a different application**

The video recording of *Citizen Kane* is used as the footage for the plot analysis and this style analysis.

2. **Reuse of objects from one instance in another instance created from a different schema, and used for a different application**

Scene objects from the plot segmentation of *Citizen Kane* are reused to investigate the proposition that ingress occurs at the start of Scenes whereas withdrawal occurs at the end of Scenes. This is the second example of this important kind of reuse.

3. **Reuse of objects in the same schema and the same application**

Ingress and Withdrawal Collections are reused in the Comparisons Temporal Collection (part of the same schema) to help investigate their purpose. This is different to the previous example of reuse as the objects are being reused in a different schema, but the same application (style analysis in this case).

In summary, this final example completes the demonstration that VCMF can support the cross section of schemes and their structural attributes and uses. It also concludes the investigation of the reuse supported by such an approach.

6.5 Summary

This chapter demonstrates that VCMF can describe real world examples of video content modelling that are considered by this thesis to be a cross section of video content models. This is not possible with other approaches. Furthermore, the examples have provided example cases of VCMF's ability to support a wide range of reuse of video, objects, and schemes. Especially important is the reuse of objects in instances created from different schemas and in different applications which was previously poorly supported. The following chapters reflect upon the demonstration and investigation of the examples and draw conclusions about the approach's relation to, and impact on, other research in the area.

Chapter 7 Reflections

This chapter reflects on the work presented in the preceding chapters. Section 7.1 reflects on the demonstrative method used in Chapter 6 to show VCMF's utility. Section 7.2 reflects on the investigation of VCMF's support for reuse of objects and schemes. Following this, Section 7.3 briefly reflects on the software toolset developed in Chapter 5 to support the use of VCMF. The next chapter reviews these reflections and suggests future work that could be undertaken to address the issues raised here. Section 7.4 compares VCMF to other work which supports abstract and concrete modelling. Finally a short summary concludes the chapter.

7.1 Reflections on Demonstrating the Applicability of VCMF

This thesis aims to demonstrate that a semantic data modelling based framework can support the definition of video content models and their schemes (see Chapter 3 for a full definition of the thesis). Chapter 6 presents three different examples of video content modelling which use markedly different schemes: plot segmentation of narrative films, interactive video-map navigation, style analysis of narrative films. These schemes are, with a few minor difficulties discussed later in this section, successfully described using VCMF. Furthermore, these examples exhibit characteristics which in total cover all the characteristics of video content models identified in Chapter 2 (see Table 12 in Chapter 6 for an overview of the characteristics exhibited by the examples). The successful description of these examples using VCMF demonstrates that it can describe a cross section of schemes and their models. Such descriptions are not currently supported by other approaches; of particular note is VCMF's ability to describe both the video-map and the plot segmentation schemes as they are radically different in characteristics and use.

From this demonstration it is posited that VCMF can describe other models and schemes from current literature as all their characteristics have been shown to be describable by VCMF. The demonstration does not provide evaluative judgements about VCMF's ability to describe schemes and models. Similarly, the problems with VCMF discussed in the following sections come from introspective insight into VCMF's use, not from rigorous evaluation. Such work is outside the scope of this thesis as it involves more examples of use and more users of VCMF. The next chapter discusses further reasons why such evaluations have not been undertaken at this stage *e.g.* the problems of separating evaluation of VCMF from its representations. Moreover, the next chapter considers the future work needed to conduct such evaluations. The key is that the framework's ability to describe different schemes and models, and to usefully relate models to their video, must be demonstrated before it is subjected to evaluation. This thesis demonstrates the utility of the framework for such descriptions; future evaluations and other work are now possible.

7.1.1 Differentiating Between Schemas

To demonstrate the framework's utility for describing schemes and models, and to facilitate investigation of the framework's support for reuse, three examples were selected from a large number of possible examples. These examples are considered to have markedly different schemes and models. This raises the issue of discriminating between schemes. In this thesis, schemas (VCMF's descriptions of schemes) which have different kinds of object and relationship types are considered to be different; the examples are distinguished because their schemas all have unique combinations of relationship and object types. In particular, the style analysis schema is the only schema to use Pre-defined and Temporal Collections. Similarly, the plot segmentation schema is unique in its use of inferred sequences in collections (Attribute-Ordered Collections). The video-map schema does not have any unique types, but it does have a unique combination of types in comparison to the other schemas.

This thesis places less emphasis on the *use* of schemas and instances, *i.e.* whether they are used for analysis and/or presentation (deterministic or non-deterministic). However, the selected examples are representative of the three main uses of models and schemes discussed in Chapter 2:

- **Analysis**: the plot segmentation and style analysis.
- **Presentation (deterministic)**: the plot segmentation.
- **Presentation (non-deterministic)**: the video-map.

In order to fully demonstrate the framework's support for different kinds of use each kind of object and relationship type needs to be shown for each kind of use. This is discussed in the next chapter.

7.1.2 VCMF's Descriptive Generality

Another issue with regard to VCMF's demonstration is that VCMF was defined to be able to describe the schemes and models identified in Chapter 2. However, it does not necessarily follow that it can describe all possible schemes. Some object or relationship types not identified in Chapter 2 might exist, or come into existence.

The strength of VCMF's approach, which gives weight to the position that it can describe other schemes and models, is its ability to:

- **Describe different kinds of relationships** (not just composition and sequence relationships or others initially defined by VCMF). The video-map highlights this by using Schema specific relationships to define spatial relationships between objects (not originally specified by VCMF).
- **Structure annotations.** VCMF uses BNF to describe the grammar of annotations. This provides a flexible mechanism for defining the structure of annotations.
- **Define the cardinality of relationships.** The specification of relationship cardinality

means that structural consistency in models can be maintained. VCMF relationships have one source and possibly multiple destinations. Combinations of relationships could be used to address the requirement of multiple sources or destinations [56] (*i.e.* one relationship for each source or destination). This again highlights the ability of VCMF to represent forms of relationships which are not part of VCMF's definition (returned to in Chapter 8).

7.1.3 The Two Aspects of VCMF's Demonstration

The demonstration of the thesis can be considered in two parts: first showing the applicability of semantic data modelling techniques to support video content modelling, and second establishing the applicability of the selected base types. These two parts are discussed in this section.

7.1.3a Applicability of semantic data modelling

This thesis demonstrates for the first time that semantic data modelling can be applied to the problems of describing video content models and schemes. Semantic data modelling research gives the basis [96] for proposing that it can describe other schemes and models. VCMF, on the other hand, provides certain building blocks (base object and relationship types) which are specific to video content modelling. This makes it less generally applicable than semantic data modelling, but ensures that it is better suited to the problems of modelling video content.

The applicability of the semantic data modelling approach is reinforced as the cross section of schemes used in the demonstration was successfully described by VCMF without the notion of inheritance provided by object-oriented approaches. Such approaches use inheritance in part to support reuse of class definitions [143]. However, the reuse of schemes (classes of models) requires descriptions of resemblances between schemes rather than the notion of inheritance. This distinction and a possible solution to the problem of reusing schemes is returned to in the next chapter.

7.1.3b Issues related to VCMF's base types

The base object types imposed by VCMF determine the schemas and instances that can be described. These base object types are suitable for describing the three examples, but from the examples, and VCMF's definition, the following issues are evident (returned to in the next chapter):

- **Clip like behaviour**

VCMF supports the most desirable form of partitioning video documents identified in Chapter 2: segments. However, some uses of instances may wish to have clip like behaviour enforced *e.g.* ensuring that shots in a plot segmentation never overlap. Systems such as Rowe *et al.*'s video database [165] need clip like behaviour as they are explicitly interested in video from a film metaphor where segment (shots) should not overlap. Other systems such as Davenport *et al.*'s ConText [53] use clips purely because of the storage mechanism employed. From their discussion there is no explicit reason for not having overlapping segments. In order to satisfy all uses of video there needs to be some mechanism in VCMF to determine whether an object represents a clip or a segment.

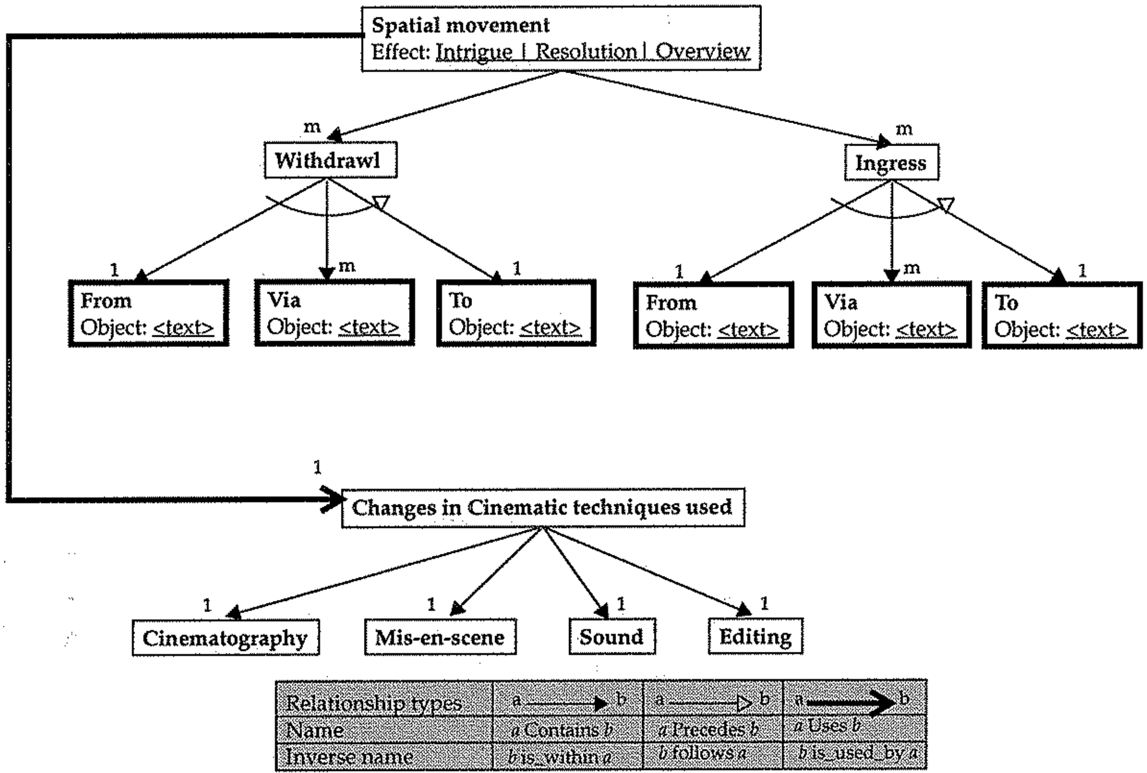
- **Temporal relationships**

  VCMF's temporal relationships can be rather awkward to use. For example, the style analysis uses a Temporal Collection to determine if certain patterns occur at the start or end of a shot. For a starts temporal relationship to hold between two objects their start_frames must be equal. In reality *fuzzy* temporal comparisons are needed [6]. These allow some *margin of error* when determining if a temporal relationship holds between objects. For example, the starts temporal relationship would hold between two objects if their start_frames are nearly equal. A problem with this solution is determining what is *nearly* equal and conversely what is not.

  This reflects a general problem with systems which define segments in terms of start and end frames. It is often the case that a particular segment may start somewhere within a set of frames *e.g.* in analysis of human behaviour captured in video it may be difficult to determine in which frame an action starts or finishes. An alternative approach to solving the problem of awkward temporal relationships would be to have *fuzzy* segments. Such segments would be composed of a beginning, middle, and end set of frames allowing these ambiguities to be represented.

- **Relationships between member objects in different collections**

  There is no explicit mechanism for defining relationship types which apply between member objects of different collections in an instance; several VCMF relationships types must be used to represent them. This is illustrated in the style analysis example of Chapter 6. To briefly recap, the schema in the example is used to support the style analysis of narrative films. Of particular interest are the stylistic patterns evident in the film (*e.g.* the spatial movement into or out of a scene) and the cinematic techniques used to create them (*e.g.* the use of editing and cinematographic techniques). In the style analysis schema the uses relationship type would be better represented if it connects a member of a Spatial movement Collection to a member of the Changes in Cinematic Techniques used Collection. Figure 58 illustrates an extract of this requirement: the thick line indicates that relationships in the instance should be instantiated between members of the Collections, not the Collections themselves. Currently relationship types can only be defined between object types and so several uses relationship types must be defined between the members of the two Collections. Figure 59 illustrates an extract of the current schema for style analysis which highlights the various uses relationship types that are defined to support the situation in Figure 58. This is a rather cumbersome way of achieving the situation in Figure 58, but again it shows that VCMF is flexible enough to support such a definition through the use of multiple relationships.



Relationship types	a ——▶ b	a ——▷ b	a ➔ b
Name	*a* Contains *b*	*a* Precedes *b*	*a* Uses *b*
Inverse name	*b* is_within *a*	*b* follows *a*	*b* is_used_by *a*

Figure 58: Extract of the style analysis schema with different uses relationships

- **Complex presentation**

VCMF does not support complex presentation such as overlapping segments, fading between segments *etc.* Algebraic video [59] supports such complex presentation in a way that could be adapted for VCMF; the composition algebra could be represented using relationship types in VCMF. This problem of complex presentation is returned to in the next chapter where a solution is proposed.

7.1.4 The Effect of VCMF's Base Object Types on its Use

The base object types imposed by VCMF affect the ways in which instances can be used. It appears from the three large examples that VCMF is more suitable for describing instances for analysis (*e.g.* the plot segmentation), non-deterministic presentation (*e.g.* the video-map) and simple deterministic presentations such as the review of Kane's life in chronological order (as opposed to complex deterministic presentation). One reason for this is that they do not rely on complex presentation such as the aforementioned overlapping of segments, fading between segments *etc.* Complex forms of presentation also occur in non-deterministic presentation. For example, ConText [53] graphically positions segments on a screen and uses overlapping and fading of segments to indicate their relative importance. This form of presentation could not currently be supported by VCMF. This is in part due to the current toolset implementation as the graphical

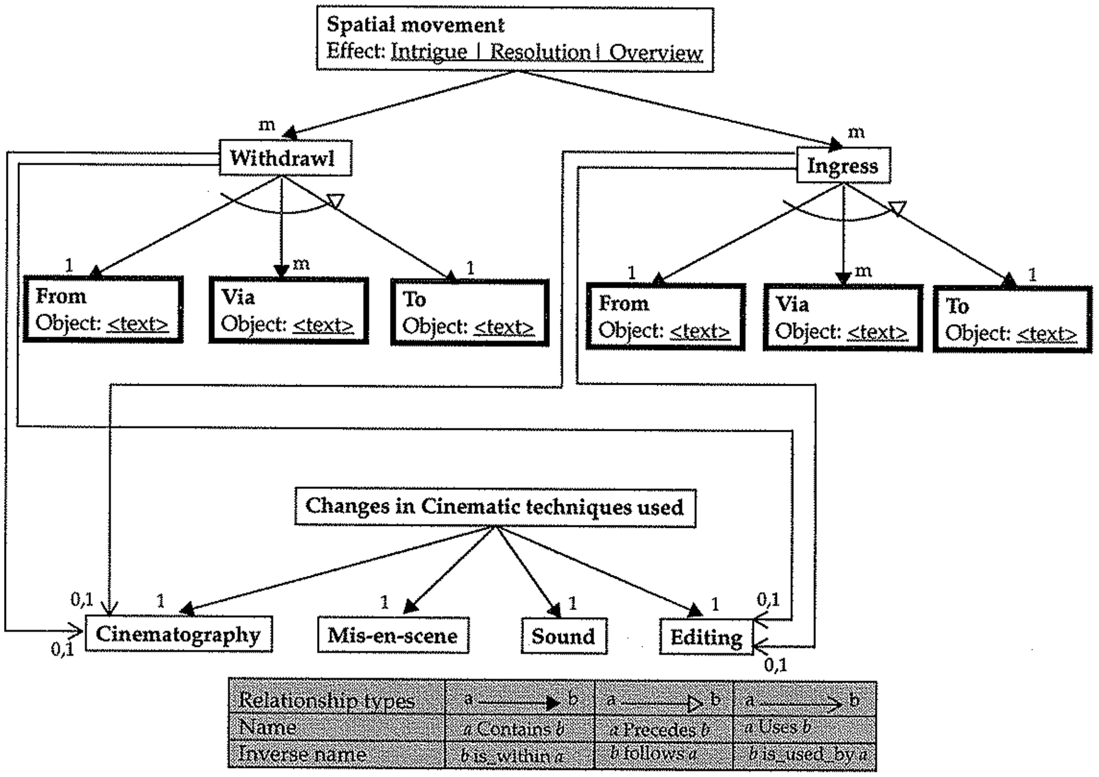


Figure 59: Extract of the current style analysis schema

positioning of segments on a screen is not part of VCMF, it is the responsibility of the toolset; objects are solely descriptive and do not perform any presentation. The advantage of this approach is that objects can be reused in different ways in different contexts without affecting the model.

Another reason for this apparent predisposition of VCMF for analytic video content models is that VCMF is based on semantic data modelling techniques. As with research in other domains, such as HDM [71], the semantic data modelling approach of VCMF means that it is more suitable for describing instances which *are* structured and regular (such as the plot segmentation which is derived from a well defined structure in academic literature), rather than instances used for *creative* presentation. In creative presentations 'the author tends to follow his feelings in a somehow unpredictable way (*e.g.* inventing new nodes [object types] and new navigation patterns [relationship types] on the fly) rather than planning his exercise in advance' [71].

7.2 Reflections on VCMF's Support for Reuse

The second part of the thesis consists of an investigation of the feasibility of VCMF's support for reuse; it identifies problems with the approach and raises issues about its future use and development. To date there has been little work which addresses semantic data modelling's support for reuse and so it is unclear what it would mean to support reuse in such an approach. Some work has been conducted in the hypermedia field with respect to the reuse of multimedia objects. So far this work has provided a taxonomy of reuse [74] which is used in this thesis, and some small examples of reuse [35][74]. The investigation in this thesis contributes to the field by exemplifying different kinds of reuse and VCMF's support for them, and by raising issues about such support for reuse. This section considers the issues of reuse related to the thesis.

7.2.1 The Definition of a Taxonomy of Reuse

The taxonomy of reuse considered in Chapter 6 is derived from work by Garzotto *et al.* [74]. It is used to identify a cross section of kinds of reuse for investigation in the thesis. In Garzotto *et al.*'s discussion of reuse they define reuse of a data item as 'a data item (simple or complex, traditional or multimedia) is used in two different applications, or in two different parts of the same application' [74]. They regard an *application* as a particular piece of software *e.g.* a word processor. However, in video content modelling it is less clear what an application is. In Chapter 6 it is considered to be a collection of schemes used for one purpose *e.g.* plot analysis. Different definitions could be proposed such as: an application is a particular use of one scheme and the model it produces *e.g.* the plot segmentation of *Citizen Kane* would be one application, and the plot segmentation of *Casablanca* would be another. Clearly this definition affects the taxonomy of reuse, and what can be said about the different kinds of reuse exhibited.

In terms of the work identified in Chapter 2, this thesis' definition of an application is the most appropriate as each system is regarded as a different application (typically in current literature one system supports one application). This means that the taxonomy clearly shows reuse between applications (systems) which Chapter 2 highlights as a major problem in current research.

In total, seven examples of reuse are presented in Chapter 2. The issues raised by these examples are discussed in the following sections. First the reuse of objects is discussed, and second the problems of reusing schemas are considered.

7.2.2 Reuse of Objects

The investigation of the reuse of objects shows that whilst some forms of object reuse are supported by VCMF, others are currently problematic or impossible. Importantly, the investigation shows that VCMF can support forms of reuse not well supported by other current research: reuse of objects between schemas by reference rather than by copying. The problems highlighted in the investigation provide valuable insight into the problems of reuse in this approach, and in general. This section first discusses the successful examples of reuse followed by the unsuccessful examples.

7.2.3 Successful Reuse of Objects

All the examples of reuse which involved reuse of objects in a different schema are successful:

- The guided-tour of the video-map (same application).
- The view of Kane's life in chronological rather than narrative order (same application).
- The comparisons between patterns of use in the style analysis and the plot segmentation (reuse between both the same and different application).
- The documentary about the video-map (different application).

This shows that VCMF, and semantic modelling in general, has the ability to support the reuse of objects between models of different schemes whilst retaining access to the original context. This is important because traditionally objects and their associated video are reused by copying. This results in the loss of their original context; retaining original context is an important aspect of reuse [74]. No previous approaches to video content modelling have exhibited this ability.

7.2.4 Unsuccessful Reuse of Objects

The nature of the investigation implies that some problems may be encountered. It was not clear before the investigation what the problems of reuse were, nor how they could be addressed. Several problems with VCMF's support for reuse are identified in Chapter 6, and discussed in this section.

7.2.4a Problems of reusing sub segments and overlaying video

The documentary about *Citizen Kane* and the reuse of objects from the narrative of *Citizen Kane* in its end credits highlight the two main problems with VCMF's support for reuse. They can both be considered partial successes as they crudely support the intended sequence of video produced by reuse. However, both examples of reuse require the reuse of *sub* segments of objects' segments and overlaying video and/or sound with the reused video.

In the end credits of *Citizen Kane* only parts of shots from the narrative are reused in the credits. Furthermore, the name of the actor who plays the character is overlaid on top of the reused video. Similarly, in the documentary about *Citizen Kane,* typically only parts of shots are reused, and these are usually overlaid with additional sound. Neither of these characteristics are currently supported by VCMF; in the crude examples of reuse whole shots are displayed rather than parts of shots, and there is no overlaying of video and/or audio. Some other work which considers the reuse of video, such as the trailer creation work of Pfeiffer *et al.* [155], also suffers from this problem. The next chapter considers some extensions to VCMF to support such characteristics of reuse.

In terms of Garzotto *et al.*'s taxonomy of reuse [74] these forms of reuse involve subtraction of (reusing part of a shot), and addition to (overlaying sound and/or images) data as it is being reused. Further investigations of reuse need to include a richer taxonomy of reuse than that used in this thesis in order to consider these forms.

7.2.4b The problem of reusing objects within a Collection

A final problem with reuse not exhibited in the examples, but evident by considering VCMF's definition, is the reuse of objects within the same Collection. Currently this is difficult as there is no direct way of specifying the linear sequence if one object appears more than once in the sequence. This is due to the nature of sequence relationships in VCMF which are similar to those of other approaches such as HDM [70] and Sawhney *et al.'s* HyperCafé [173]. One solution to this problem would be to have a richer definition of sequence, though the next chapter discusses how it can be addressed using the current VCMF definition, highlighting the utility of VCMF's approach. The identification and solution of this problem provide useful insights for similar work.

7.2.5 Reuse of Schemes

The main investigation of reuse concentrates on the feasibility of reusing objects and their associated video (identified in Chapter 2 as being an important form of reuse in video content modelling). A secondary investigation considers the reuse of schemes (represented in VCMF by schemas). This section reflects on the reuse of schemas in the examples.

In Chapter 6 there are problems with each example of schema reuse. These problems all stem from the same fundamental problem (shared by other research which support the reuse of scheme like descriptions) and are discussed in the following paragraphs.

First, it is not possible to reuse the plot segmentation schema to construct the documentary. This is because the plot segmentation schema is designed for *analysis* of narrative plot in an already constructed video. The members of collections in the schema are ordered on their start_frames which is ideal for analysis in this case. However, in order to *construct* the documentary the user creates a model from different video segments which are not usually in the correct order already. Therefore the schema needs to have collections whose members are ordered by the user, not by their start_frames. This results in modification of the plot segmentation schema to support the documentary's construction.

Similarly, there are problems when the documentary schema is reused to create the documentary about the video-map. In this case object types (Flow and Turn Units) need to be added to the schema to allow inclusion of objects from the video-map.

There needs to be some way of supporting the use of modified versions of schemas without having to create copies of existing schemas and modifying them. This would also allow modifications in schemas to be propagated to schemas that are derived from them. The next chapter considers a solution to this problem in detail.

7.3 Reflections on Software Support for VCMF

This thesis is concerned with demonstrating the applicability of semantic data modelling techniques to the problems of video content modelling. It is not concerned with the problems of constructing video content modelling systems *per se*, but a system was constructed to support VCMF. This section briefly reflects upon the issues raised in such an undertaking.

The system discussed in Chapter 5 is not necessarily the most efficient, nor is it easy to use. However, it does fulfil its role of supporting the demonstration of the use of VCMF to model large exemplar video content models and to relate these models to video and schemes. Moreover, it shows that VCMF has been specified in sufficient detail to be realised in a software system. Its shortcomings mean that it does not support all of the advantages of VCMF. In particular, it does not enforce the BNF rules for attribute values, it does not automatically calculate temporal relationships, and it does not enforce relationship constraints and cardinality. However, even with these shortcomings the toolset shows that it is feasible to construct a system for video content modelling based on VCMF, and that such a toolset can support use and reuse of models and schemes.

7.3.1 The Toolset's Support for the Benefits of Semantic Modelling

In Chapter 4 three advantages of Garzotto *et al.*'s [71] schema based approach to hypermedia design are raised and considered with respect to VCMF. These are: the ability to check for completeness, the consistent use of relationships, and the automatic generation of relationships. The toolset realises two of these benefits. First, as discussed in Chapter 5 the completeness check is supported, although it is very slow and so is not interactively available. Second, relationships between members of a Collection are automatically instantiated and maintained by the toolset. This removes from the user a significant burden involved in creating semantic data models. However, the toolset does not enforce consistent use of relationships as there is currently no constraint or cardinality functionality implemented.

7.3.2 The General Problem of Labour Intensive Video Analysis

Using the toolset highlights one of the main problems of current video content modelling; the amount of effort it takes to create a video content model. This is not a problem caused by the VCMF approach nor the toolset themselves. It is a problem caused by the large amount of data that needs to be analysed to create a model; constructing models containing hundreds of objects and relationships is time consuming and tedious [153]. Currently VCMF and the software system have no support for automatic analysis (segmentation and annotation) of video which can reduce the time and effort needed to construct models. The next chapter considers how automated analysis could be supported to the best advantage of the VCMF approach.

7.3.3 User Comprehension of Schemes and Models

A general problem of systems which support the description of schemes and models is: how can a user grasp the correspondence between the scheme and the model. In hypermedia approaches such as hypermedia templates [34] there is an emphasis on the description of graphical layout at the scheme level. The relationship between graphics defined by the scheme and model is relatively easily visualised by showing the general layout with reference to the scheme, and the specific layout, graphics, and text with reference to the model. It is harder for users to perceive other correspondences such as relationships between objects, or how object and relationship types' constraints will affect their use [72].

The VCMF toolset addresses this problem in part by providing a consistent user interface between the scheme and model tools. Objects and object types are represented in similar ways. Similarly, relationships and their types are visually represented in similar ways in both tools. However, one object type looks much like another and so all the objects of an instance tend to look similar; identifying their object type requires matching the name or ID of the object type.

Furthermore, there needs to be a high level of integration between the scheme and model manipulation so that changes in one are quickly and obviously reflected in the other. This supports more iterative scheme authoring where schemes may be refined as models are developed *e.g.* the iterative development of coding schemes for analysis of observational data [170]. This moves approaches such as VCMF away from being most suitable for well defined and static modelling schemes (as discussed previously), but raises its own issues discussed in the next sub-sections.

7.3.4 Task Specific Presentation Issues

Models, and parts of models, may be reused for different tasks. This leads to the general issue of how to present models in task specific ways. In some user interface development approaches (see Kovacevic [113] for a review) and some video content modelling approaches such as VideoSTAR [92] this is typically tackled by separating the model from its presentation which allows different presentations of the same model. VCMF follows this approach by allowing different components of the software toolset to communicate with the Application Interface. Indeed, the implementation approach of VCMF is based upon that of VideoSTAR. The major advantage of VCMF is that it supports the definition of schemas whereas VideoSTAR uses a fixed implicit scheme. As discussed throughout this thesis, using fixed schemes restricts a system to one application and limits reuse of segments and their annotations between systems. Interestingly, the object-oriented approach of HyperStorm [16] also separates the presentation of the model from the model itself. This is interesting because the responsibility for presentation usually lies with the objects in object-oriented approaches.

The Video-map Navigation component provides a good example of the separation VCMF supports between the model and its presentation. The video-map instance can be manipulated and navigated through using the generic toolset components such as the Simple Navigation component discussed in Chapter 5. However, the Video-Map Navigation component provides a user interface which is tailored to the task of navigating through the video-map which reduces the effort a user has to make to navigate through it. At this point it is worth noting that VCMF itself has no concept of the different tasks that video content models and schemes are put to; the user interfaces provide task specific behaviour.

7.3.5 Propagating Changes in Schemes

The issue of close integration between scheme and model editing is raised by Catlin *et al.*'s hypermedia templates [34] and in terms of databases by Bouneffa *et al.* [22] and Kim *et al.* [110]. Catlin *et al.* discuss *hot* templates and consider what happens to a model if its scheme changes. In hypermedia templates nothing happens to models after their scheme is changed. The same is true of instances created from schemas in VCMF. This is a problem of change propagation (also referred to as *schema evolution* [22][110]) which needs careful consideration. It might be the case that the author wants instances to reflect the change in their schemas. On the other hand they may be changing a schema for one particular instance, but not want the other instances of the schema to be updated. Bouneffa *et al.* describe various possible solutions and implementation approaches to solving this problem.

It is worth making the distinction between this issue and that of change propagation between schemas discussed in Section 7.2.5. This section is concerned with propagating changes in schemas to instances realised from them; the problem raised in Section 7.2.5 relates to the propagation of changes in schemas to schemas derived from them.

7.4 Comparisons to Other Work

In relation to other video content modelling work VCMF is unique. It is the only approach tailored to handling video content models which also supports the description of their schemes. When comparing VCMF with other work in the field it is not the user interface which is of interest, but the underlying data model; how the video content models are structured, and what semantics they have.

The demonstration in Chapter 6 shows that VCMF can describe a cross section of video content modelling schemes and their models. This means that VCMF can describe the underlying data model of other approaches in the field. Moreover, VCMF can support the creation and manipulation of such video content models. The three examples that constitute the cross section involve the use of schemes for plot segmentation, spatial structuring for video-maps, and cinematic style analysis. In terms of current video content modelling systems, the structure of plot segmentation is reflected in most work.

Systems such as Rowe *et al.*'s Video Database Browser [165], Hjelsvold *et al.*'s VideoSTAR [92], and many others all use similar plot based structures (though they are typically simpler than the plot segmentation structure used in this thesis). The use of spatial structuring is more unusual. Systems such as Lippman's movie maps [121] and Sawhney *et al.*'s HyperCafé [173] both use similar spatial structuring to that exemplified in this thesis. Finally, the style analysis scheme is related to systems interested in annotating video with the cinematic techniques used (*e.g.* Media Streams [55]), and those which analyse patterns of behaviour captured in observational data [169].

User interfaces for video content modelling are a different matter altogether. As discussed previously, the VCMF toolset is intended to facilitate VCMF's demonstration; it provides poor user interfaces in comparison to other work in the field. However, the aim of the thesis is not to investigate video content modelling user interfaces. The next chapter speculates on how VCMF could be used to support the evaluation of user interfaces for video content systems.

7.4.1 Comparisons to Hypermedia Work

Because of VCMF's uniqueness in the field of video content modelling it is more revealing to compare VCMF to hypermedia approaches than other video content modelling approaches. In particular, it is useful to compare VCMF to hypermedia approaches which tackle the problem of describing schemes and models of dynamic media (which video can be considered a member of).

In comparison to hypermedia design frameworks such as HDM [70], VCMF is more concerned with the actual creation and manipulation of models and schemes. HDM provides a framework for designing applications, but these applications must be created using a software system such as Hypercard [9]. This means that there is a gap between the design and the implementation which must be bridged manually. However, their emphasis on design rather than implementation means that design methodologies have been developed to structure the design process. No such methodologies have been developed with VCMF.

HB1 [175] is a hyperbase which uses the semantic data modelling approach. Instead of concentrating on design as HDM does, HB1 is concerned with the generation of hypermedia applications. In this respect it is similar to VCMF's concentration on generation of video content models rather than design methodologies. However, HB1 can not support descriptions of video content as VCMF does; it concentrates on static media and relationships between them. Importantly though, HB1 identifies the ability to create relationships between applications as beneficial. VCMF uses a similar approach when supporting reuse of video and associated annotations between applications.

VCMF also differs from approaches such as HyperStorm [16] which are concerned with the creation and manipulation of models and schemes. HyperStorm is an object-oriented approach whereas VCMF takes a semantic data modelling approach. Although

HyperStorm uses some ideas from semantic data modelling such as the description of semantic relationships between objects, VCMF more closely meets the requirements of video content modelling identified in this thesis. This is primarily because object-oriented approaches are concerned with modelling behaviour and attributes of objects whereas semantic data modelling approaches are concerned with attributes only. For video content modelling only the modelling of attributes is important; behaviour is captured in the video itself. Therefore modelling behaviour adds an unnecessary level of description. Furthermore, the semantic data modelling concepts used in HyperStorm are expressed using objects. For example, relationships are represented by objects. This means that such concepts are not fundamental to the approach; they are handled indirectly leading to less powerful description and manipulation.

7.5 Summary

This chapter reflects on the issues associated with VCMF raised in previous chapters. Four broad areas of issues are considered: the demonstration of VCMF's utility, the investigation of VCMF's support for reuse of objects, the investigation of the reuse of schemes, and the utility of the toolset. Each issue provides insights into the utility of the VCMF approach and semantic data modelling in general. Furthermore, raising these issues contributes an understanding of the issues in such a research domain and will help to lay the ground for future research and development. Requirements for solutions to problems are flagged in this chapter and returned to in the next which takes a more holistic view of the problems and their solutions.

Chapter 8 Conclusions

This chapter concludes the thesis. First it presents the contribution of the thesis; how the work surveyed in Chapter 2 and Chapter 3 has changed as a result of this thesis. This is followed by solutions to the problems identified in the previous chapter. Future evaluation of VCMF, as well as evaluations and investigations supported by VCMF are then presented. These are followed by a discussion of how VCMF could be integrated into other frameworks. Finally the chapter and the thesis itself are summarised.

8.1 Concluding the Thesis

In concluding the thesis this chapter performs four functions:

1. It presents the contribution of the thesis.
2. It explores future development to be carried out on, and using, VCMF.
3. It suggests evaluations that can be undertaken of, and using, VCMF.
4. It suggests investigations that can be undertaken of, and with, VCMF.

The important aspect of this chapter is the dual role of VCMF. Not only does this chapter suggest how VCMF can be developed and evaluated, but it also suggests how VCMF can be used to develop, investigate, and evaluate other research. This dual role is a result of the utility of VCMF itself; it has been shown that it can describe video content models and their schemes and so it can now be used to support research in a range of domains with interests in video, not just video content modelling.

8.2 Contribution

This thesis makes several contributions to the research discussed in Chapter 2 and Chapter 3. In general terms it shows the applicability of semantic data modelling in a previously unexplored domain: video content modelling. Previous work has concentrated on the use of such modelling techniques for describing hypermedia applications [70][175]. Although video content modelling can be considered part of the hypermedia research field, there has previously been no investigation of the applicability of such techniques to the particular problems of video content modelling.

This chapter also contributes to the semantic data modelling field as it raises the notion of family resemblance to support related versions of schemes and their derivation. This highlights an aspect of semantic data modelling which needs further investigation.

In terms of video content modelling research this thesis contributes a demonstration of the ability of VCMF to describe schemes and models, and to use these with their related video. This addresses the first goal of this thesis outlined in Chapter 1: to demonstrate that one computer based framework can describe the structures and semantics used by systems dealing with video. This demonstration changes the research covered in Chapter 2 as previously it was not possible to support such descriptions. Typically the structural regularities of models used by applications were implicit and embedded in applications themselves. Now one framework (VCMF) can be used to describe the schemes and models of all work in Chapter 2, and to support the use of these models and their related video. As discussed previously, Chapter 6 shows VCMF's ability to describe three different schemes: plot segmentation, video-map, and film style analysis. Chapter 7 describes how these schemes explicitly represent implicit structures of current systems.

This thesis also contributes an investigation of the feasibility of VCMF's support for reuse of parts of models and their schemes, and the problems that this causes. In general, the reuse of parts of models has been given little attention in semantic data modelling compared to other approaches such as the object-oriented paradigm where some consider it to be a central tenant [156]. Some uses of semantic data modelling for hypermedia such as HDM [70] and HB1 [175] have provided small examples reuse. However, this work is unique in its consideration of: reuse of video and associated annotations by reference, and reuse of schemes. This investigation meets the second goal of the thesis outlined in Chapter 1: to investigate the forms of reuse supported by the framework (VCMF).

Furthermore, VCMF supports the reuse of objects between models whilst retaining access to their original context. Previously this accessibility was difficult to support as different approaches could not understand each other's models and schemes. This resulted in video being reused by copying *via* an intermediate format such as a QuickTime [10] file or a video cassette. This increases the storage space needed as well as reducing the ability to locate the video's original context. It also implicitly encourages the use of clips which, as discussed in Chapter 2, provide impoverished logical segmentation of video documents compared to segments.

The framework itself contributes a platform from which further research can be conducted. Moreover, future research domains include those outside the scope of this thesis *i.e.* the contributions of this thesis open it up to new domains of research. Later sections in this chapter consider future work including various forms of evaluation (not only of VCMF) that can now be undertaken using VCMF *e.g.* evaluation of HCI issues related to different uses of video content models. These discussions themselves contribute to understanding of what research topics are salient in the area.

As discussed in Chapter 2, that chapter itself contributes to current research. It provides a review of current literature in terms of two key paradigms in video content modelling: the conventional (film based) paradigm, and the new (non-linear) paradigm. Furthermore, it reviews work from different disciplines, which use video for different uses, and use different processes to manipulate video and models. Previous reviews of video content modelling typically concentrated on particular uses of video, or particular paradigms of video usage.

8.3 Future Development of VCMF

The first area of future work considered in this chapter is the development of VCMF in light of the issues raised in the previous chapter. Several developments are discussed in this chapter, each of which addresses several unresolved matters. The problems themselves are not peculiar to VCMF; they typically come from the approaches that VCMF was developed from. Therefore the proposed solutions have a bearing on other research which tackles similar issues.

8.3.1 Family Resemblance

The problems associated with reusing schemes (*e.g.* the attempted reuse of the plot segmentation to construct a documentary about *Citizen Kane* and an area of the video-map) are symptomatic of the underlying approach of VCMF (semantic data modelling): VCMF lacks the ability to describe *family resemblance* (inspired by Rosch *et al.* [164]). In this thesis family resemblance means the ability to take one schema, make a new version of it with some amendments, and have some mechanism which defines the resemblance between the two schemas. Currently schemas can be copied and changed for a new purpose, but there is no notion of how schemas are related. Providing such relationships between schemas supports both description of the resemblance between schemas, and a mechanism for the propagation of changes in one schema to schemas derived from it. Such support is of most benefit to schema authors. It provides them with an explicit definition of the similarity and grouping of schemas, a naming mechanism, and a mechanism which can support propagation of change through the family. Other approaches which are also based on semantic data modelling would also benefit from the inclusion of family resemblance.

Family resemblance is not object inheritance [143]. Object inheritance means that one class of objects takes properties and behaviour from a super class, and can modify them. In fact, the previous chapter suggested that object inheritance's utility to video content modelling is limited; family resemblance provides a technique for defining family relationships which is more powerful than object inheritance and more related to the problems of using of video content schemes (and their models). Similarly, family resemblance is not *schema evolution* [22][110] which involves refining one schema and ensuring the correct update of sub classes and their objects. Family resemblance concerns the derivation of one schema from another with some modifications so that they are not identical; a schema consists of many object and relationship types. In this way family resemblance is more akin to *schema versions* [110]. However, there are typically only two schema versions: *transient* and *working*. Family resemblance involves the definition of many related *versions* of the same schema; the family members.

The problem of family resemblance has not been addressed in other hypermedia research. This is primarily because in other domains schemas are seldom reused, and if they are they are reused without modification. For example, HDM [70] and HB1 [175] support the definition of classes of hypermedia applications, but typically only one instance is created from each schema. In these approaches the schema acts as a template to structure an application's development. On the other hand, Hypermedia Templates [34] do explore the reuse of templates (similar to schemas), but do not consider different versions of the same template and the resemblances between them.

8.3.1a Family resemblance in the examples used in this thesis

The issue of family resemblance is raised in the very first schema definition. In the plot segmentation schema the Scene object type has a unique attribute called Approximate date which takes a date value. This attribute is present to support the modelling of flashbacks in films. However, not all films contain flashbacks and so this attribute is only of use to a subset of films including films such as *Citizen Kane* and *Casablanca*. Using family resemblance this could be accounted for by defining a schema for plot segmentation in general, and defining a particular family member for plot segmentation involving flashbacks.

Figure 60 illustrates the possible family resemblances of schemas in this thesis. At the root of the family is the general structure of plot in narrative films described by Bordwell *et al.* [21] (the general structure is shown in the grey box). This provides an abstract definition that Films are composed of Parts which are composed of Sub-Parts and Scenes *etc.* From this schema two other schemas are defined (definition shown by the arrowed lines). These are used for the analysis of plot and the construction of presentations. They define different orderings of members of collections: start_frame ordering for analysis and user ordering for presentation (shown in italics in the figure).

In this thesis the presentation schema is used to create the documentary about *Citizen Kane* (illustrated in the figure to the right of the schema). From the analysis schema another is derived which includes support for the description of flashbacks using a date attribute in the Scene object type. This is used to create the original plot segmentation of *Citizen Kane* and is then refined to allow the reuse of objects from the narrative in the end credits. The third example of reuse of the plot segmentation in Chapter 2 uses this schema to construct its model of *Citizen Kane*'s plot segmentation. The presentation schema is refined to include object types from the video-map, and then used to create the documentary about the area captured in the video-map.

Figure 60 is a simple example of family resemblance. Moreover it is a single interpretation of the family resemblance of schemas considered in this thesis. Another way to structure the resemblance might consider kinds of films first (here whether the film has flashbacks or is a documentary), and then the use of the schemas (here whether it is for plot analysis or presentation construction) rather than the use of schemas first and then the kind of film it supports. This is an issue of categorisation which is not only

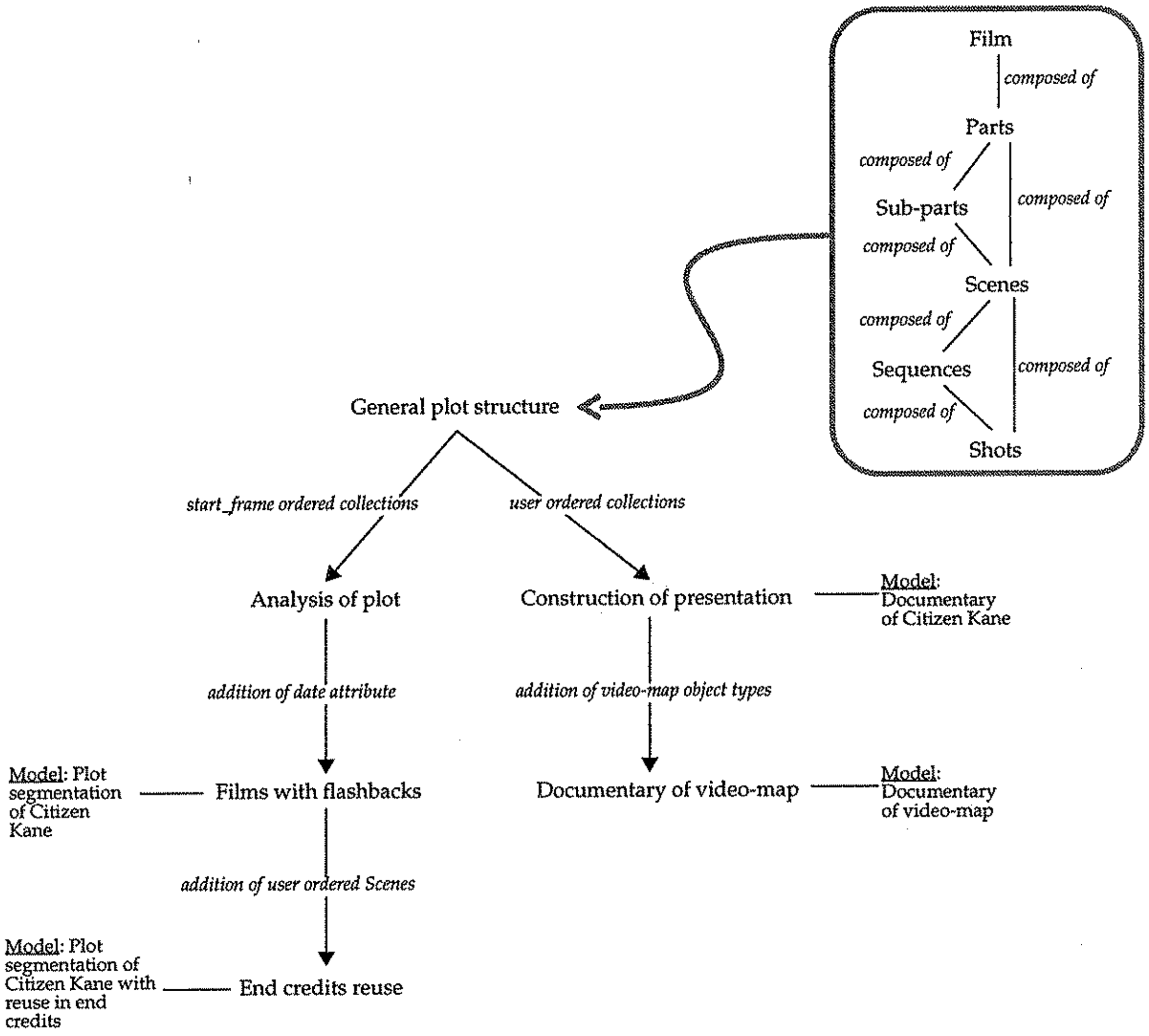

Film
composed of
Parts
composed of
composed of
Sub-parts
composed of
Scenes
composed of
composed of
Sequences
composed of
Shots
General plot structure
start_frame ordered collections
user ordered collections
Analysis of plot
Construction of presentation
Model: Documentary of Citizen Kane
addition of date attribute
addition of video-map object types
Model: Plot segmentation of Citizen Kane
Films with flashbacks
Documentary of video-map
Model: Documentary of video-map
addition of user ordered Scenes
Model: Plot segmentation of Citizen Kane with reuse in end credits
End credits reuse

Figure 60: Possible family resemblance of schemas

evident in family resemblance, but also in other areas such as object inheritance. Furthermore, this problem increases with each additional family member. For example, the general plot structure (of films) in Figure 60 could be considered to be part of a family of plot in general which has other family members including plot structure in theatre, books *etc*. The issue is how to usefully categorise members and show their family resemblance.

In these examples family resemblance has been considered *between* categories (where each category only has one schema member). It is also usefully employed *within* categories (where there are several members of a category and there are resemblances between them). For example, there are resemblances between the schemas used for analysis of plot and construction of presentations as they are both derived from the same schema (the schema for general plot structure). These resemblances are *within* the category of schemas derived from the general plot structure.

8.3.2 Change Propagation Between Schemas

An issue concerned with family resemblance is how change is propagated through the family. For example, what happens to the schema for analysis of films with flashbacks if an additional object type is added to the schema representing the general structure of plot.

In VCMF this issue could be addressed by ensuring that modifications in schemas are always propagated to their instances and related schemas. In this approach schemas which are changed for only one instance are regarded as a new family member (a new schema of the family). This means that changes in the schema only affect one instance. Whereas changes to the original schema (for simplicity, consider the root of the family) are propagated through all instances and schemas created from it. This problem and solution clearly require careful consideration when implementing systems which support descriptions of schemes and models. Addressing this issue of change propagation addresses the problems of change propagation for schemas *and* instances discussed in the previous chapter.

8.3.3 Support for Complex Presentation and Reuse

Chapter 7 suggested that VCMF is most suited to uses which do not involve complex presentation of video. This is partly due to VCMF's current lack of support for such presentation. Chapter 7 also identified problems with reusing objects when the video needed to be altered in some way *e.g.* overlaying sound/ video (titles are overlaid on video reused in the end credits of *Citizen Kane*), or reusing parts of segments (parts of shots are reused in the *Citizen Kane* documentary).

These three problems could be addressed by extending VCMF's base relationship types to support complex composition in a manner similar to Algebraic video's composition operations [59]. Table 21 illustrates the composition operations which could be usefully represented by new VCMF base relationship types, and how these base relationship types would solve the two problems of reuse highlighted here. Moreover, providing such base relationship types would support complex (deterministic) presentations.

The provision of an is_parallel_to relationship type would support the overlay of two segments by displaying them in parallel. For example, Shots reused in the end credits would be related to segments containing text by the is_parallel_to relationship and so would be presented at the same time. Similarly, provision of an intersects_with relationship type could support reuse of sub segments *i.e.* only reusing those parts of the segment that intersect with the given segment. For example, to support the reuse of sub segments of Shots in the end credits the Shot would be reused, but additionally it would be related to a small blank segment which defines the sub segment's temporal extent using the intersects_with relationship. When presented the video that intersects the two

Algebraic Video's Composition Operations	New VCMF Base Relationship Type	Particular Reuse Problem Addressed
concatenation	concatenates_with	
union	union_with	
intersection	intersects_with	May be suitable to address problem of reusing a sub segment of video.
difference	is_different_to	
parallel	is_parallel_to	Overlay of one segment with another *e.g* actors' names and extra sound.
conditional	n/a (handled by constraints)	
loop	n/a (handled by looping collections)	
stretch	n/a	
transition	fades_to	
contains	n/a (handled by queries)	

Table 21: Possible translation of video algebra's composition operations [59] into VCMF relationships to address reuse problems

temporal extents will be played *i.e.* the required sub segment of the Shot.

An alternative approach to supporting these two problems of reuse would be to infer which segments reuse the same video rather than explicitly reusing objects in new contexts *via* relationships. Auguierre Smith *et al.*'s [12] stratagraph allows the user to infer which video is common between segments and hence reused. However, one of the basic premises of semantic data modelling (and so VCMF) is that relationships should be explicit; this alternative approach would not be in keeping with the rest of the VCMF approach. Moreover, this solution would only be suitable for analysis of reuse rather than construction of presentations *via* reuse.

8.3.4 Support for Clip Behaviour

The previous chapter identified VCMF's current lack of clip like behaviour which is desirable for some uses of video (*e.g.* Rowe *et al.*'s video database [165]). In VCMF this behaviour could be supported by defining some mechanism whereby members of the same collection should not temporally overlap, and moreover should abut temporally. Such tight temporal constraints would mean that these objects would typically be used for analysis purposes; they may make definition of clip like segments easier as it can be assumed that they will abut and not overlap. Moreover, such constraints may make automated definition of segments (with clip like behaviour) easier.

A simple way to solve this problem for VCMF would be that all objects have an attribute which determines whether it has clip or segment behaviour. Alternatively a new base object type could be defined which represents clips rather than segments. According to the definition of clips in Chapter 2 they cannot share frames and so are atomic (similar to Units). As the responsibility for enforcing clip like behaviour lies with the toolset a simpler approach would be to have a mode of the toolset which enforced clip like behaviour. However, this approach reduces the descriptive power of the model and would mean that all objects in a model had to either be clips or segments.

8.3.5 Attribute Value Specification

As mentioned in the previous chapter, possible attribute values are defined by BNF rules. This means that only textual annotations can be used. Some uses of video content models require other media as annotations *e.g.* storage of audio and graphics in annotations [15]. There are several possible solutions to this problem including:

- Creating BNF rules which represent different media. For example, the rules <audio> and <graphic> could be defined to allow audio and graphics in annotations. These graphics and audio could then be stored in the annotation, or referenced in some other repository. In fact, the current VCMF definition and toolset supports referencing of video frames in annotations *e.g.* to support representative frames in annotations the frame number of the representative frame is stored rather than the frame itself. The problem with defining such BNF rules for other media is that there is no notion of the structure of the audio or graphics; the role of the BNF rules is to prescribe the structure of annotations.
- Using a different specification language such as SGML [26]. Again, the problem with such approaches is that they define the structure of textual information, not other multimedia data.
- A more radical solution is to integrate VCMF with a hypermedia framework. This would then naturally support multimedia annotations of segments. Such a proposition is returned to in Section 8.7.

8.3.6 Extending Constraints for Relationship Types

In VCMF schemas relationship types hold between two object types and constraints can be placed on the number of destination objects in the instance. This does not directly support multiple kinds of destination objects nor multiple source objects. This is because VCMF's treatment of relationships is based on work by De Rose [56] which treats relationships in a similar way. Currently multiple relationship types must be used when multiple kinds of destination object, or more than one source object is required. A solution to this would be to make the constraints of relationship types richer, for example storing a list of possible object types and their cardinality rather than just one possible object type. Similarly, a constraint list could be specified for the source of the relationship type.

However, such an approach makes relationships themselves much more complicated, and some of their characteristics become implicit within their cardinality list rather than being explicitly between two relationship types. Moreover, they may be harder to present to the user as there needs to be some way of showing that one relationship has several source and destination objects types, as well as showing the cardinality of these relationships. This might involve displaying these complex relationships in the same way as the current approach would: as separate relationship types between object types.

The current approach to defining relationship types is sufficient for VCMF's purposes; it is simple and easily understood. Moreover, more complex relationships can be represented by a collection of current relationships. This shows that approaches to describing relationships such as De Rose's [56] are good solutions to the problem as they are powerful enough to be combined to describe other relationships.

Similarly, the current definition of VCMF does not support the definition of relationship types which can be instantiated to members of the destination collection (*e.g.* the Uses relationship type of the style analysis schema discussed in the previous chapter). The current solution can be cumbersome as multiple relationship types must be set up to represent the intended relationship type. One solution to this is that relationship type constraints could be extended to indicate that the destination object in the instance is a member of the collection rather than the collection itself. As with the previous suggested addition to constraints this could lead to confusion about the meaning of relationships (as some meaning would be implicit), and may also be presented in the conventional manner anyway. Again, the ability of VCMF to represent these kinds of relationships shows its descriptive utility.

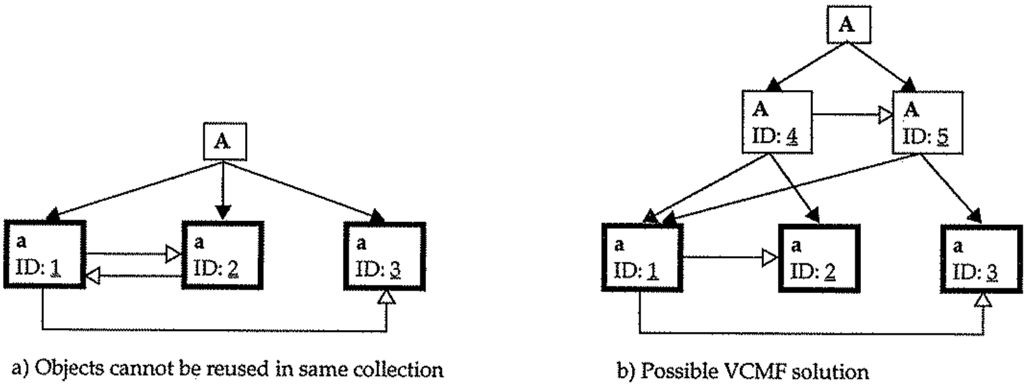
8.3.7 Functional Content Modelling

As highlighted in Chapter 2, VCMF is not concerned with *functional* modelling of video content *i.e.* modelling the function of segments. This is because current video content models are primarily interested in modelling the descriptive structure and semantics of models of video content.

The flexibility of the VCMF approach means that such functional modelling could be supported by describing the function of objects or relationships. For example the relationships used in Aquanet [132] to define the function of nodes of information could be supported by defining Schema specific relationships in VCMF with appropriate names and constraints to reflect their function.

8.3.8 Richer Sequence Relationships; Supporting Reuse of Objects Within a Collection

The previous chapter discussed the problem of reusing objects within a Collection. This problem arises in VCMF and similar approaches because sequential base relationship types simply state that one object follows another. Therefore if an object is reused in the same Collection there will be two sequential relationships leading from it to its two possible destination objects. For example, VCMF could not directly describe a Collection with objects a1, a2, a3 in the sequence a1 → a2 → a1 → a3 (object a1 is reused). Figure 61 illustrates this problem. In Figure 61a there is no way of telling which object comes after a1, it could be a2 or a3. Figure 61b illustrates a possible solution to the problem using VCMF: A contains A4 and A5, A4 precedes A5 therefore the sequence of objects in A is a1 → a2 → a1 → a3 as required.



a) Objects cannot be reused in same collection

b) Possible VCMF solution

Relationship types	a ——▶ b	a ——▷ b
Name	*a* Contains *b*	*a* Precedes *b*
Inverse name	*b* is_within *a*	*b* follows *a*

Figure 61: Problem of reusing objects in the same collection and possible VCMF solution

The current solution is rather cumbersome. An improvement would be to provide a richer form of sequence relationship. For example, using an extra attribute in sequence relationships to include some notion of the linear position it represents. Other approaches which use a similar form of sequential relationship (*e.g.* hypervideo [173]) also need to be refined with this problem in mind. A new approach to this problem could be developed and tested using VCMF. Results of this development could then be used to modify other approaches, though re-implementing other approaches using VCMF would provide more flexible applications with more potential for reuse.

8.4 Furthering the Demonstrative Approach

The demonstrative approach used in this thesis provides a first demonstration of the utility of VCMF and semantic data modelling in the video content modelling domain. Further demonstrations need to build upon this work to show different aspects of VCMF's utility in more detail.

One such demonstration is the exhibition of VCMF's support for different kinds of uses of schemas. Considering the *use* of schemas and instances requires at least three examples of each object and relationship type; one for analysis, another for deterministic presentation, and another for non-deterministic presentation. This requires at least three times as many examples because usage is not currently considered in example selection. These examples could all be constructed and tested using the current VCMF framework; the main problem is identifying enough suitable examples to demonstrate VCMF's support for different kinds of use.

It is worth considering whether every type of object and relationship *can* be used in the three different ways. From the demonstrations in Chapter 6 it seems that some base object types such as Temporal Collections are of most use for analysis as members are related in terms of their temporal extent. This point is returned to in Section 8.6.1 which considers the investigation of the relationship between base types and their use. At this stage it would be more judicious to investigate whether there is a relationship between base types and their use rather than trying to demonstrate that all VCMF base types are appropriate for all uses.

8.5 Future Work: Solving Toolset Problems

Most toolset problems are soluble through programming effort; they are not intractable. The time scale of the thesis and the desire to concentrate on modelling issues rather than implementation issues led to the implementation being sufficient to demonstrate the utility of the thesis, but not as complete as the full VCMF definition requires. As discussed in the previous chapter, only the BNF rules for attribute values, the automatic calculation of temporal relationships, and the relationship constraints and cardinality were not implemented. Other parts were implemented, but could be improved by refining the algorithms to provide quicker response times.

8.5.1 Supporting Automated Analysis

One area not supported by the current definition of VCMF or its toolset is that of automated analysis of video, in particular the automated definition of segments, annotations, and relationships. This is completely out of the scope of this thesis due to the size and complexity of this problem. However, now that VCMF has been demonstrated it would be useful to be able to develop automated techniques which could help create models and hopefully reduce model creation time.

As discussed in Chapter 2, current automated analysis is very domain and application specific *e.g.* automated shot detection is only suitable for videos using a film paradigm. An important issue is how to describe the criteria for segmentation in a domain independent manner. Gabbe *et al.* [66] addresses this issue in a simple manner by allowing the definition triggers including: audio triggers, computer output triggers, and shot detection triggers. When a trigger's condition is met, for example a person speaks, the video is partitioned (segmented). This approach provides a simple means of defining segments which can be applied as the video is captured, so saving later segmentation time. However, this approach is limited in its applicability because it does not provide a generally applicable mechanism for defining triggers; each trigger is a special case. Hampapur *et al.* [85] define models of shot boundaries which allow them to be automatically detected and classified. This provides more flexible segmentation in terms of shot boundary detection, but does not support any other forms of segmentation. More flexible mechanisms need to be developed which can infer some description of events such as a speaker raising their voice, or a different camera being used. These descriptions can then be used as the basis of more flexible segmentation. Furthermore, such descriptions will aid the automatic analysis of video content which is currently limited to visual feature detection. Typical current automated analysis systems produce annotations which have limited semantic content; they reflect some visual characteristic of the video rather than its meaning in the context.

One way forward would be to include support for various forms of audio and visual constraints in VCMF object descriptions. These could then be used during analyses to constrain which kind of objects and descriptions could be generated, and when segments could have their start and end frames defined. VCMF already provides some support for constraining which objects could exist within different parts of video (this can be used in a similar way to the state transition networks of Swanberg *et al.*'s work [186]). In particular, Pre-Ordered Collections can be used to define possible sequences of object types in a video. These Collections could be used to reduce the possible number of object types that could be extracted from a video at a particular time. For example, such Collections can ensure that after a News title object has been created, only News item or News reel objects can be created. The reduced number of possible object types and their audio and video constraints should then make the automated analysis more tractable and less error prone (as exemplified by the work of Swanberg *et al.* [186]).

8.6 Future Work: Evaluations and Investigations

As discussed in Chapter 6 and Chapter 7 no rigorous evaluations are carried out in this thesis because VCMF's utility for describing schemes and models needs to be demonstrated before its use can be evaluated. Now that VCMF's utility has been shown future work can consider evaluations of VCMF, and investigations into its use and future uses. These evaluations and investigations have impact not only on VCMF, but also on other research which uses similar techniques. Moreover, as the utility of VCMF has been shown it can be used to support evaluations and investigations of topics unrelated to VCMF itself. This section considers all these forms of evaluation and investigation. First it speculates on the evaluation of VCMF itself. Then Section 8.6.3 broaches HCI issues related to video content modelling. This is followed by consideration of implementation issues, and finally the utility of reuse.

8.6.1 Evaluating VCMF Itself

In order to evaluate VCMF there must be a separation of the evaluation of VCMF and its representation (in this thesis there are two representations used: the graphical notation and the user interface representation). This is a major difficulty with evaluating VCMF and is one of the reasons that such evaluations are not undertaken in this thesis. Another major problem is the lack other approaches to compare VCMF with (for comparative evaluations).

This section considers three kinds of evaluations of VCMF itself: how well it describes models and schemes, how useful it is, and how usable it is. The topic of usability is returned to in Section 8.6.3 which considers the usability of interfaces for video content modelling.

8.6.1a Evaluating VCMF's efficacy

From the demonstration it is not clear whether the cross section of models and schemes had to be coaxed into a VCMF representation, or whether VCMF is suitably tuned to the kinds of descriptions necessary. An evaluation needs to consider how well VCMF describes these and other schemes and models. In order to do this some measure of the amount of effort required to describe a scheme and model needs to be defined *e.g.* the number of times VCMF descriptions have to be changed, though this also reflects on users' understanding of VCMF (and the usability of the representations, though this should be evaluated separately as discussed in Section 8.6.3). A problem with such an evaluation is that there are no similar approaches which VCMF can be compared to.

Evaluating VCMF's ability to describe models and schemes also needs to consider how well VCMF describes models and schemes intended for different purposes. Specifically, whether is it more suitable for one kind of modelling than another. Currently it appears that its weak point is in describing complex presentations, but this may be due to the lack of support for such presentations (addressing this problem is discussed in Section 8.3.3). Once complex presentation is part of VCMF a number of different kinds of schemes and models need to be constructed using VCMF; the efficiency of description of different kinds of models can then be compared *i.e.* whether a lot of effort expended to make VCMF describe models with certain uses.

Similarly, assessments need to examine whether the base type of an object affects its potential use. Investigations need to be undertaken to explore the use of base types; each base type needs to be used for each possible purpose (in real world examples). There may be base types for which certain uses are not appropriate. For example, Temporal Collections appear to be of most use in analysis, investigations need to establish whether they are suitable for other uses of video content models.

8.6.1b Evaluating VCMF's utility and usability

Fundamentally, an evaluation needs to establish whether VCMF is useful. In particular, whether it saves time and effort compared to similar approaches. One way of evaluating this would be to describe the same application in VCMF and another video modelling system and measure factors such as the time taken and errors made when using the different approaches. The two major difficulties of evaluating an approach such as VCMF are: first it will be difficult to separate the evaluation of VCMF's utility from its representation's utility (*e.g.* the utility of the toolset), second, there are currently no similar approaches to compare VCMF with. Furthermore, the use of automated analysis software (*e.g.* shot boundary detection algorithms [49]) will clearly affect the utility of an approach and so must be taken into account.

Although there are no similar approaches which VCMF could be compared to, it could be compared to current single application systems discussed in Chapter 2. The models and schemes of single application systems could be implemented in VCMF and comparisons made between the time and effort required to learn to use, and construct models and schemes in the two approaches. An important factor is the amount of learning time required; each approach will require time to learn, but if VCMF takes longer to learn than single systems, the number of systems that can be learnt in the same time needs to be established. This will determine the threshold above which it is quicker to learn and use VCMF than multiple single systems. Again, it is important to separate the evaluation VCMF from the user interfaces provided to the user.

Related to such evaluations is the assessment of other people's use of VCMF. All the models and schemes used in this thesis were created by one person (the author of VCMF). An evaluation needs to establish whether other people can easily use VCMF, or whether the concepts and techniques involved are too complex or time consuming to master. In other words, whether the approach chosen met its intended aim which was to provide an accessible framework. To address this several users could be instructed on the use of VCMF and then asked to complete several tasks including the construction of models and schemes and supporting reuse between models. Various metrics for evaluation include: number of errors made, time taken to complete tasks compared to an expert, whether the users expressed dislike for VCMF. Note again that evaluation of VCMF must be separated from its representations *e.g.* the toolset used in the evaluations.

8.6.2 Evaluating Semantic Data Modelling's Efficacy for Video Content Modelling

Another form of evaluation now possible is the separate evaluation of the utility of semantic data modelling in the domain of video content modelling. Such an evaluation needs to concentrate on whether the concepts of semantic data modelling are appropriate for the domain. The rationale behind choosing semantic data modelling as the approach in Chapter 3 is that it appears to be more suitable for the requirements of VCMF. This evaluation needs to establish whether this is this true, and what problematic constraints semantic data modelling places on VCMF that could otherwise be avoided.

8.6.3 HCI Issues

As well as evaluating VCMF separately from its representations, the representations themselves should be evaluated. These could be evaluated in terms of criteria such as user comprehension, ease with which they are learnt, ease of use, and expressiveness.

The work in this thesis purposefully avoided any concern for the human computer interaction involved in using video content models. Now that the framework's utility has been shown it can be used to support the evaluation of many of the HCI issues that exist in interacting with video content models (not just VCMF). The section covers a few of the HCI related issues that this framework could be used to evaluate.

The user comprehension of the relationship between schemas and instances presented by the toolset needs to be evaluated. In particular, whether the current approach is easily understood by users. Different user interface approaches (using the same underlying VCMF systems) could be used in an experimental design to determine which fosters the greatest understanding of the link between schemes and models. Measurements could include time to complete tasks related to models and schemes, and testing the user's understanding of the relationship between the schemes and models. Results of such an evaluation are of interest to other systems which describe both schemes and models *e.g.* Catlin *et al.*'s hypermedia templates [34].

Further use of VCMF to support evaluation of HCI issues related to video content manipulation could include assessment of users' understanding of the meanings of relationships between video segments. For instance, VCMF could be used to support evaluation of aesthetic and rhetorical aspects of relationships between video segments as discussed by Liestøl [119]. In particular, VCMF can contribute the ability to support evaluations of different forms of relationships presented and used in different ways in different kinds of models, and for different purposes.

In terms of VCMF itself, the toolset has no concept of the task being performed, and VCMF has no notion of what tasks models and schemes are designed to support. An investigation could consider what relationship is there between the tasks schemes and models are used for, and their description and representation by the toolset. An example of the currently *implicit* relationship between task and representation is the video-map which shows the use of a task specific user interface to support navigation. Further work needs to consider how such relationships can be made explicit, whether it would be useful to do so, and what such relationships could be used for. Such an investigation would contribute to the future design of video content modelling user interfaces, and the understanding of how one model or scheme can be usefully used for different tasks.

8.6.4 Evaluation of Implementation Issues

The main implementation issue of VCMF that needs to be evaluated is: whether the use of semantic data modelling is better than other approaches such as object-oriented modelling. Objects in semantic data modelling perform no processing and so rely on some external processing to display them. This was selected for VCMF as it supports more flexible reuse and the objects required are purely descriptive (perform no processing). Objects in other approaches such as object-oriented modelling (*e.g.* [167]) are descriptive *and* perform processing. They typically contain functionality to present themselves rather than relying on external processing. So, for the definition of VCMF semantic data modelling is the best approach, but in terms of implementation an assessment of whether another approach would have been better needs to be undertaken.

Evaluation of this issue requires a system comparable to VCMF, but which uses objects that perform processing. The same model and scheme could then be created and manipulated using both systems and the performance characteristics (*e.g.* processing time, storage space required) compared. The results of such an evaluation would have impact on other systems which address the problems of describing abstract and concrete models for example HyperStorM [16] and HDM [72].

8.6.5 Evaluation of Reuse

The feasibility of VCMF's support for reuse has been investigated in this thesis. This raised issues which are important to other approaches which support reuse. Once support for reuse has been successfully demonstrated (for example using future versions of VCMF which include the solutions proposed in this chapter) it can be evaluated. As with other forms of evaluation, it is important to separate the evaluation of VCMF's support for reuse from evaluating its representation.

The fundamental aspect of reuse that will need to be evaluated is the cost/ benefit ratio of reuse itself (rather than the system that supports it), and reuse in an approach such as VCMF. Such costs and benefits depend on the application and the type of footage required. They include: time, effort, financial cost, storage space, quality *etc.*

Evaluations of the benefits of reuse may involve comparisons to the costs of capturing new footage as reuse is intended to avoid such capture. An issue here is the selection of evaluative metrics which allow suitable comparisons to be made; both reuse and capture have their own sets of costs and benefits, the difficulty is in usefully comparing them. A complication with such comparisons is the kind of footage required; different kinds of footage have different sets of costs and benefits for reuse and capture.

8.7 Future Use of VCMF: Integrating VCMF into Other Frameworks

With the increase in processing power of computer systems there have been many hypermedia systems which integrate video with other media such as text and graphics [31][176]. As discussed previously, the fundamental problem with using such hypermedia approaches to support video content modelling is that they regard video as an atomic unit of information and so do not support suitable descriptions of its content.

VCMF does support such description of video, but does not support descriptions of other media. This raises the question of whether it would be useful to integrate such an approach into a hypermedia framework to enhance its video modelling. Bulterman [31] discusses the importance of being able to reuse video clips within hypermedia applications, and even between applications. However, CMIF [31] only supports video clips with no associated annotations. Supporting segments of video and more descriptive modelling would support more flexible reuse of video.

A fundamental issue concerning the integration of video content modelling into hypermedia frameworks is how the models are described. Two possibilities exist:

1. **Separation.** The video content model and the hypermedia model could be supported separately. Video would then be regarded by the hypermedia framework as a unit of information which has its own internal structure. This weakens the integration of video into the framework and may cause synchronisation and reuse problems between the two frameworks.
2. **Integration.** The hypermedia models and video content models could be described using the same framework. For example, HDM [70] could be extended to support the description of video content. Video would then be regarded as an entity containing: units (HDM's notion of hypertext nodes) representing annotations, relationships, and segments related to annotations. A problem with this approach is that hypermedia frameworks do not usually consider modelling the content of data *e.g.* the content of a graphical image. They are more concerned with descriptions of the presentation of data *e.g.* HyperStorM [16] is primarily concerned with describing the presentation dynamics of multimedia objects. Conversely, video content modelling is concerned with modelling the *content* of video. As such it may be that neither approach can be suitably extended to support modelling of hypermedia presentation and video content.

8.8 Summary

This chapter suggests how many of the issues raised in the previous chapter can be addressed by future work. It also suggests many pieces of research that can be carried out using VCMF *e.g.* HCI evaluations of user interfaces for video. In this way VCMF has opened the doors for future research by providing a framework for modelling video whose descriptive utility has ben demonstrated.

This thesis contributes the following:

- A demonstration of the applicability of semantic data modelling techniques in a previously unexplored domain: video content modelling.
- Suggestions about how family resemblance could be integrated into semantic data modelling to improve support for schema use and reuse
- The definition of VCMF which can describe video content models, their schemes, and

relate them to video. Such descriptions were not previously possible.

- An investigation of the feasibility of using VCMF to support reuse of parts of models, and schemes. Especially important is the reuse by reference of parts of models in models realised from other schemes which was not previously possible.
- An exploration of the problems and possible solutions to using such an approach to describe models and schemes as well as to support reuse.
- A survey of current literature from a broader perspective than previously attempted.

Two of the main suggestions for addressing the issues identified in this thesis are:

- **Family resemblance.** Large scale use of VCMF and other semantic data modelling based approaches will necessitate using many different schemas. Without support for family resemblance it will be difficult to manage the many derivations of schemas that will be required. Even in this thesis a family of six members was proposed for schemas involving plot structure. Considering the number of different schemas needed for all the systems discussed in Chapter 2 the need to support grouping and classification of schemas is clear.
- **Presentation.** VCMF's ability to construct sequences is currently limited to simple sequences. There needs to be some development which supports complex presentation and sequencing of segments including descriptions of fades, parallel presentation *etc.* This chapter identifies a possible approach to this problem by expressing the composition operations of video algebra [59] using VCMF's relationships.

To briefly recap, this thesis defines a framework, shows its ability to describe video content models and schemes, and investigates its support for reuse. It raises issues with the approach and suggests solutions, many of which are applicable to other research domains. Furthermore, the framework itself can now be used to support other research concerned with the modelling of video content.

References

1 Adah, S., Candan, K. S., Chen, S-S., Erol, K., & Subrahmanian, V. S. (1996). The Advanced Video Information System: Data Structures and Query Processing. *Multimedia Systems, Volume 4, Number 4*, pp. 172-186.

2 Adam, J. F., & Tennenhouse, D. L. (1994). The Vidboard: A Video Capture and Processing Peripheral for a Distributed Multimedia System. *Proceedings of ACM Multimedia '93, Anaheim, CA*, pp. 113-120.

3 Adiba, M. (1996). STORM: An Object-oriented Multimedia DBMS. *Multimedia Database Systems: Design and Implementation Strategies, Nwosu. K. C., Thuraisingham, B., & Berra, P. B. (eds.)*, pp. 47-88.

4 Adobe Systems Incorporated. (1985). *Postscript Language; Tutorial and Cookbook.* Addison-Wesley Publishing Company.

5 Albarn, D. (1993). For Tomorrow. *Modern Life is Rubbish.* EMI Records Ltd.

6 Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM, Volume 26, Number 11*, pp. 832-843.

7 AM Newswire. (1998). *DVD Insider - Premastering Reference.* http://www.dvdinsider.com/pr-author.html.

8 Aoki, H., Shimotsuji, S., & Hori, O. (1996). A Shot Classification Method of Selecting Effective Key-Frames for Video Browsing. *Proceedings of ACM Multimedia '96, Boston, MA, USA*, pp. 1-10.

9 Apple Computer Inc. (1987). *HyperCard User's Guide.* Cupertine, C. A.

10 Apple Computer Inc. (1991). *Proposal to Standardise a Temporal Media Movie Format.* Cupertine, C. A.

11 Arman, F., Depommier, R., Hsu, A., & Chiu, M-Y. (1994). *Proceedings of ACM Multimedia '94, San Francisco, CA USA*, pp. 97-103.

12 Auguierre Smith, T. G., & Pincever, N. C. (1991). Parsing Movies in Context. *Proceedings of USENIX '91, Nashville, TN*, pp. 157-168.

13 Avid Technology (1997). *How To Choose a Digital Nonlinear Editing System.* http://www.avid.com/news/publications/whitepapers/how_to_choose.html.

14 Avid Technology (1997). *Workflow in the Digital Newsroom.* http://www.avid.com/news/publications/whitepapers/workflow.html.

15 Baecker, R., Rosenthal, A. J., Friedlander, N., Smith, E., & Cohen, A. (1996). A Multimedia System for Authoring Motion Pictures. *Proceedings of ACM Multimedia '96, Boston, MA USA*, pp. 31-42.

16 Bapat, A., Wäsch, J., Aberer, K., & Haake, J. M. (1996). HyperStorM: An Extensible Object-Oriented Hypermedia Engine. *Proceedings of Hypertext '96, Washington DC, USA*, pp. 203-214.

17 Beacham, F. (1995). Movie of the Future: Storytelling with Computers. *American Cinematographer, April 1995*, pp. 36-48.

18 Bell, A. (1995). Walk on Water. *Tarantula*. EMI Publishing Ltd.

19 Birtwistle, G., Dahl, O., Myhrtag, B., & Nygaard, K. (1973). *Simula Begin*, Auerback Press, Philadelphia.

20 Bobick, A. F. (1993). Representational Frames in Video Annotation. *Proceedings of the 27th Annual Conference on Signals, Systems and Computers, IEEE Press*, pp. 111-114.

21 Bordwell, D., & Thompson, K. (1993). *Film Art, An Introduction* (Fourth Edition). McGraw-Hill Inc.

22 Bouneffa, M., & Boudjlida, N. (1995). Managing Schema Changes in Object-Relationship Databases. *Proceedings of OOER '95: Object-Oriented and Entity-Relationship Modelling, Gold Coast, Australia*, pp. 113-122.

23 Bowie, D. (1977). Sound and Vision. *Low*. EMI Records Ltd.

24 Brown, M., Foote, J., Jones, G., Jones, K., & Young, S. (1995). Automatic Content-Based Retrieval of Broadcast News. *Proceedings of ACM Multimedia '95, San Francisco, California*, pp. 35-43.

25 Bruns, G. (1988). *Germ: A Metasystem for Browsing and Editing*. MCC Technical Report STP-122-88, Austin, Texas.

26 Bryan, M. (1988). *SGML An Author's Guide to the Standard Generalised Mark-up Language*. Addison-Wesley Publishers Ltd.

27 Bryan-Kinns, N. J. (1998). Modelling Video in Hypermedia. *Proceedings of Euromedia '98, Leicester, England*, pp. 105-111.

28 Bryan-Kinns, N. J. (1998). VCMF: A Framework for Video Content Modelling. *To appear in: Multimedia Tools and Applications*.

29 Bryant, S. (1989). *The Television Heritage, Television Archiving Now and in an Uncertain Future*. BFI Publishing.

30 Buchanan, M. C., & Zellweger, P. T. (1992). Specifying Temporal Behaviour in Hypermedia Documents. *Proceedings of ACM ECHT, Milano*, pp. 262-271.

31 Bulterman, D. C. A. (1995). Embedded Video in Hypermedia Documents: Supporting Integration and Adaptive Control. *ACM Transactions on Information Systems, Volume 13, Number 4*, pp. 440-470.

32 Cappo, M., & Darling, K. (1996). Measurement in Motion. *Communications of the ACM, Volume 39, Number 8*, pp. 91-93.

33 Casner, S. (1994). Are You on the MBone? *IEEE Multimedia, Summer 1994*, pp. 76-79.

34 Catlin, K. S., & Garrett, L. N. (1991). Hypermedia Templates: An Author's Tool. *Proceedings of Hypertext '91*, pp. 147-160.

35 Cavallaro, U., Garzotto, F., Paolini, P., & Totaro, D. (1993). HIFI: Hypertext Interface for Information Systems. *IEEE Software, November 1993*, pp. 48-41.

36 Chen, P. P. (1976). The Entity-Relationship Model - Toward a Unified View of Data. *ACM Transactions on Database Systems, Volume 1, Number 1*, pp. 9-36.

37 Cherfaoui, M., & Joly, P. (1993). A Survey of Automatical Tools for the Content Analysis of Digital Video. *Report available via FTP from: ftp.irit.fr/private/svideo.ps.*

38 Cherfaoui, M., & Bertin, C. (1994). Two-stage Strategy for Indexing and Presenting Video. *Proceedings of Storage and Retrieval for Image and Video Databases II Conference, SPIE Volume 2185*, pp. 174-184.

39 Cheyer, A., & Julia, L. (1998). MVIEWS: Multimodal Tools for the Video Analyst. *Proceedings of ACM International Conference on Intelligent User Interfaces, San Francisco, CA USA*, pp. 55-62.

40 Chow, M. D. (1989). The Role of the Video Professional in a Research Environment. *ACM SIGCHI Bulletin, Volume 21, Number 2*, pp. 83-87.

41 Christel, M., Kanade, T., Maudlin, M., Reddy, R., Sirbu, M., Stevens, S., & Watclar, H. (1995). Informedia Digital Video Library. *Communications of the ACM, Volume 38, Number 4*, pp. 57-58.

42 Christel, M. G., Smith, M. A., Taylor, C. R., & Winkler, D. B. (1998). Evolving Video Skims into Useful Multimedia Abstractions. *Proceedings of CHI '98, Los Angeles, CA USA*, pp. 171-178.

43 Chua, T-S., Lim, S-K., & Pung, H-K. (1994). Content-based Retrieval of Segment Images. *Proceedings of ACM Multimedia '94, San Francisco, CA USA*, pp. 211-218.

44 Chua, T-S., & Ruan, L-Q. (1995). A Video Retrieval and Sequencing System. *ACM Transactions on Information Systems, Volume 13, Number 4*, pp. 373-407.

45 Conklin, J., & Begeman, M. L. (1988). *gIBIS: A Hypertext Tool for Exploratory Policy Discussion.* MCC Technical Report Number STP-082-88, Austin, Texas.

46 Cowie, P. (Eds.). (1971). *A Concise History of the Cinema.* A. Zwemmer Ltd.

47 Cruz, G., & Hill, R. (1994). Capturing and Playing Multimedia Events with STREAMS. *Proceedings of ACM Multimedia '94, San Francisco, CA USA*, pp. 193-200.

48 Csinger, A., & Booth, K. S. (1994). Reasoning about Video: Knowledge-based Transcription and Presentation. *Proceedings of the Twenty Seventh Annual Hawaii International Conference on System Sciences, IEEE*, pp. 599-609.

49 Dailianas, A., Allen, R. B., & England, P. (1995). Comparison of Automatic Video Segmentation Algorithms. *Proceedings of SPIE 2615, Photonics East 1995: Integration Issues in Large Commercial Media Delivery Systems, Bellingham, WA*, pp. 2-16.

50 Davenport, G. (1989). Elastic Movies: Dynamic Links and Cinematic Browsers. *Proceeding of Computer World '89*, pp. 37-41.

51 Davenport, G., Smith, T., & Pincever, N. (1991). Cinematic Primitives for Multimedia. *IEEE Computer Graphics and Applications, July 1991*, pp.67-74.

52 Davenport, G., Evans, R., & Halliday, M. (1993). Orchestrating Digital Micromovies. *Leonardo, Volume 26, Number 4*, pp. 283-288.

53 Davenport, G., & Murtaugh, M. (1995). ConText Towards the Evolving Documentary. *Proceedings of Multimedia '95, CA USA*, pp. 381-377.

54 Davis, H., Hall, W., Heath, I., Hill, G., & Wilkins, R. (1992). *MICROCOSM: An Open Hypermedia Environment for Information Integration.* Technical Report, University of Southampton: CSTR 92-15.

55 Davis, M. (1993). Media Streams: An iconic Language for Video Annotation. *Telektronikk 4.93: Cyberspace, Volume 89, Number4*, pp.59-71.

56 DeRose, S. (1989). Expanding the Notion of Links. *Hypertext '89 Proceedings*, pp. 249-257.

57 Dimitrova, N., & Golshani, F. (1994). Rx for Semantic Video Database Retrieval. *Proceedings of ACM Multimedia '94, San Francisco, CA USA*, pp. 219-226.

58 Diskin, Z., & Cadish, B. (1995). Variable Sets and Functions Framework for Conceptual Modelling: Integrating ER and OO via Sketches with Dynamic Markers. *Proceedings of OOER '95: Object-Oriented and Entity-Relationship Modelling, Gold Coast, Australia*, pp. 226-237.

59 Duda, A., Weiss, R., & Gifford, D. (1994). Content-Based Access to Algebraic Video. *Available from: The Programming System Research Group, MIT Laboratory for Computer Science.*

60 Eysenck, M. W., & Keane, M. T. (1995). *Cognitive Psychology.* Psychology Press.

61 Fischer, S., Lienhart, R., & Effelsberg, W. (1995). Automatic Recognition of Film Genres. *Proceedings of Multimedia '95, CA USA*, pp. 295-368.

62 Fisher, C., & Sanderson, P. (1993). Exploratory Sequential Data Analysis: Traditions, Techniques and Tools. *SIGCHI Bulletin, Volume 25, Number 1*, pp. 34-40.

63 Fountain, A. M., Hall, W., Heath, I., & Davis, H. C. (1990). MICROCOSM: An Open Model For Hypermedia With Dynamic Linking. *Proceedings of the European Conference on Hypertext, INRIA, France*, pp. 298-311.

64 Frolich, D., Drew, P., & Monk, A. (1994). Management of Repair in Human-Computer Interaction. *Human Computer Interactions, Volume 9*, pp. 385-425.

65 Frisse, M. E., & Cousins, S. B. (1989). Information Retrieval from Hypertext: Update on the Dynamic Medical Handbook Project. *Proceedings of Hypertext '89*, pp. 199-212.

66 Gabbe, J. D., Ginsberg, A., & Robinson, B. S. (1994). Towards Intelligent Recognition of Multimedia Episodes in Real-Time Applications. *Proceedings of ACM Multimedia '94, San Francisco, CA USA*, pp. 227-236.

67 Gaines, B. R., & Shaw, M. L. G. (1993). Open Architecture Multimedia Documents. *Proceedings of ACM Multimedia '93, CA USA*, pp. 137-146.

68 Gall, D. (1991). MPEG: A Video Compression Standard for Multimedia Applications. *Communication of the ACM, Volume 34, Number 4*, pp. 46-58.

69 Garnham, A. (1988). *Artificial Intelligence, An Introduction.* Routledge & Kegan Paul, London and New York.

70 Garzotto, F., Paolini, P., & Schwabe, D. (1991). Authoring-in-the-large: Software Engineering Techniques for Hypertext Application Design. *Proceedings of the 6th IEEE International Workshop on Software Specification and Design*, pp. 193-201.

71 Garzotto, F., Paolini, P., & Schwabe, D. (1993). HDM - A Model-Based Approach to Hypertext Application Design. *ACM Transactions on Information Systems, Volume 11, Number 1*, pp. 1-26.

72 Garzotto, F., Mainetti, L., & Paolini, P. (1993). Navigation Patterns in Hypermedia Databases. *Proceedings of the 26th IEEE International Conference on System Sciences, MAUI*, pp. 370-379.

73 Garzotto, F., Mainetti, L., & Paolini, P. (1995). Hypermedia Design, Analysis, and Evaluation Issues. *Communication of the ACM, Volume 38, Number 8*, pp. 74-86.

74 Garzotto, F., Mainetti, L., & Paolini, P. (1996). Information Reuse in Hypermedia Applications. *Proceedings of Hypertext '96, Washington DC, USA*, pp. 93-104.

75 Geißler, J. (1995). Surfing the Movie Space: Advanced Navigation in Movie-Only Hypermedia. *Proceedings of Multimedia '95, CA USA*, pp. 391-400.

76 Gibbs, S. (1991). Composite Multimedia and Active Objects. *Proceedings of ACM OOPSLA '91*, pp. 97-112.

77 Gibbs, S., Breiteneder, C., & Tsichritzis, D. (1993). Audio/Video Databases: An Object-Oriented Approach. *Proceedings of the 9th International Conference on Data Engineering, IEEE Press, Piscataway, N.J.*, pp. 381-390.

78 Ginsberg, A. (1993). A Unified Approach to Automatic Indexing of Information Retrieval. *IEEE Expert Systems, October 1993*, pp. 46-56.

79 Goldberg, A. & Robson, D. (1989). *Smalltalk-80: The Language*, Addison-Wesley, Reading, MA.

80 Gong, Y., Chuan, C. H., Yongwei, Z., & Sakauchi, M. (1996). A Generic Video Parsing System with a Scene Description Language (SDL). *Real-Time Imaging, Volume 2*, pp. 45-59.

81 Gosling, J., & McGilton, H. (1996). *The Java Language Environment*. http://java.sun.com/docs/white/langenv/.

82 Grønbaek, K., & Trigg, R. H. (1996). Toward a Dexter-based Model for Open Hypermedia: Unifying Embedded References and Link Objects. *Proceedings of Hypertext '96, Washington DC, USA*, pp. 149-160.

83 Halsaz, F. G., Moran, T. M., & Trigg, R. H. (1987). Notecards in a Nutshell. *Proceedings of the ACM CHI+GI Conference, Toronto*, pp. 45-52.

84 Hamakawa, R., & Rekimoto, J. (1993). Object Composition and Playback Models for Handling Multimedia Data. *Proceedings of ACM Multimedia '93, CA, USA*, pp. 273-281.

85 Hampapur, A., Jain, R., & Weymouth, T. (1994). Digital Video Segmentation. *Proceedings of ACM Multimedia '94, San Francisco, CA USA*, pp. 357-364.

86 Hardman, L., Bulterman, D. C. A., & Van Rossum, G. (1993). The Amsterdam Hypermedia Model: Extending Hypertext to Support Real Multimedia. *Multimedia, Volume 5, Number 1,* pp. 47-69.

87 Harrison, B. L., (1991). Video Annotation and Multimedia Interfaces: From Theory to Practice. *Proceedings of the Human Factors Society, 35th Annual Meeting,* pp. 319-323.

88 Harrison, B., & Baecker, R. (1992). Designing Video Annotation and Analysis Systems. *Graphics Interface '92,* pp. 157-166.

89 Harrison, B. L., Owen, R., & Baecker, R. M. (1994). Timelines: An Interactive System for the Collection and Visualisation of Temporal Data. *Proceedings of Graphics Interface '94,* pp. 141-148.

90 Herrtwich, R. G. (1990). Time Capsules: An Abstraction for Access to Continuous-Media Data. *Proceedings of the 11th Real-Time Systems Symposium, Lake Buena Vista, FL.,* pp. 11-20.

91 Hibino, S., & Rundenstainer, E. A. (1996). A Visual Multimedia Query Language for Temporal Analysis of Video Data. *Multimedia Database Systems: Design and Implementation Strategies, Nwosu. K. C., Thuraisingham, B., & Berra, P. B. (eds.),* pp. 123-159.

92 Hjelsvold, R., Langørgen, S., Midtstraum, R., & Sandstå. (1995). Integrated Video Archive Tools. *Proceedings of Multimedia '95, CA USA,* pp. 283-293.

93 Hjelsvold, R., Midtstraum, R., & Sandstå, O. (1996). Searching and Browsing a Shared Video Database. *Multimedia Database Systems: Design and Implementation Strategies, Nwosu. K. C., Thuraisingham, B., & Berra, P. B. (eds.),* pp. 89-122.

94 Ho-Shing, H., & Chan, S-L. (1997). Hypertext-Assisted Video Indexing and Content-based Retrieval. *Proceedings of Hypertext '97, Southampton, UK,* pp. 232-233.

95 Hsu, W., Chua, T. S., & Pung, H. K. (1995). An Integrated Colour-Spatial Approach to Content-based Image Retrieval. *Proceedings of Multimedia '95, CA USA,* pp. 305-313.

96 Hull, R., & King, R. (1987). Semantic Database Modelling: Survey, Applications, and Research Issues. *ACM Computing Surveys, Volume 19, Number 3,* pp. 201-260.

97 Ince, D. C. (1988). *An Introduction to Discrete Mathematics and Formal System Specification*. Claredon Press, Oxford.

98 Isakowitz, T., Stohr, E. A., & Balasubramanian, P. (1995). RMM: A Methodology for Structured Hypermedia Design. *Communication of the ACM, Volume 38, Number 8,* pp. 34-44.

99 Isenhour, J. P. (1975). The Effects of Context and Order in Film Making. *AV Communication Review, Volume 23, Number 1,* pp. 69-80.

100 ISO. (1997). MPEG-7: Context and Objectives (v.3). *ISO/IEC JTC1/SC29/WG11 N1678. Available at http://drogo.cselt.stet.it/mpeg/.*

101 Jain, R., & Hampapur, A. (1994). Metadata in Video Databases. *SIGMOD RECORD, Volume 23, Number 4,* pp. 27-33.

102 Johnson, E. R. (1997). Frameworks = (Components + Patterns). *Communications of the ACM, Volume 40, Number 10,* pp. 39-42.

103 Jordan, D. S., Russell, D. M., Jensen, A-M. S., & Rogers, R. A. (1989). Facilitating the Development of Representations in Hypertext with IDE. *Proceedings of Hypertext '89,* pp. 93-104.

104 Kael, P., Mankiewicz, H. J., & Welles, O. (1985). *The Citizen Kane Book.* Methuen, London.

105 Kahn, P., & Haan, B. J. (1991). Video in Hypermedia: The Design of Intervideo. *Visual Resources, Volume VII,* pp. 353-360.

106 Kaindl, H., & Snaprud, M. (1991). Hypertext and Structured Object Representation: A Unifying View. *Proceedings of Hypertext '91,* pp. 345-358.

107 Katseff, H. P., & Robinson, B. S. (1994). Predictive Prefetch in the Nemesis Multimedia Information Service. *Proceedings of ACM Multimedia '94, San Francisco, CA USA,* pp. 201-209.

108 Kazman, R., Al-Halimi, R., Hunt, W., & Mantei, M. (1996). Four Paradigms for Indexing Video Conferences. *IEEE Multimedia, Spring 1996,* pp. 63-73.

109 Kent, W. (1978). *Data and Reality: Basic assumptions about data processing revisited.* North Holland Publishing Company, 1978.

110 Kim, W., Ballou, N., Chou, H-T., Garza, J. F., & Woelk, D. (1989). Features of the ORION Object-Oriented Database System. In Kim, W., & Lochovsky, F. H. (Eds.), *Object-Oriented Concepts, Databases, and Applications* (pp. 251-282). ACM Press.

111 King, R. (1989). My Cat is Object-Oriented. In Kim, W., & Lochovsky, F. H. (Eds.), *Object-Oriented Concepts, Databases, and Applications* (pp. 23-30.). ACM Press.

112 Koegel, J. F., Rutledge, L. W., Rutledge, J. L., & Keskin, C. (1993). HyOctane: A HyTime Engine for an MMIS, *Proceedings of ACM Multimedia '93, CA USA,* pp. 129-136.

113 Kovacevic, S. (1992). A Compositional Model of Human-Computer Dialogues. In Blattner, M. M., & Dannenberg, R. B. (Eds.), *Multimedia Interface Design* (pp. 373-404). ACM Press.

114 Krueger, C. W. (1992). Software Reuse. *ACM Computing Surveys, Volume 24, Number 2,* pp. 131-183.

115 Lans van der, R. F. (1988). *Introduction to SQL.* Addison-Wesley Publishing Company.

116 Le Gall, D. (1991). MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM, Volume 34, Number 4,* pp. 46-58.

117 Lienhart, R. (1996). Automatic Text Recognition for Video Indexing. *Proceedings of ACM Multimedia '96, Boston, MA USA,* pp. 11-20.

118 Lienhart, R., Pfeiffer, S., & Effelsberg, W. (1997). Video Abstracting. *Communications of the ACM, Volume 40, Number 12,* pp. 55-62.

119 Liestøl, G. (1994). Aesthetic and Rhetorical Aspects of Linking Video in Hypermedia. *Proceedings of ECHT '94*, pp. 217-223.

120 Lindblad, C. J., Wetherall, D. J., & Tennenhouse, D. L. (1994). The VuSystem: A Programming System for Visual Processing of Digital Video. *Proceedings of ACM Multimedia '94, San Francisco, CA USA*, pp. 307-314.

121 Lippman, A. (1980). Movie-Maps: An Application of the Optical Videodisc to Computer Graphics. *ACM SIGGRAPH, July 1980*, pp. 32-42.

122 Little, T. D. C., & Ghafoor, A. (1990). Synchronisation and Storage Models for Multimedia Objects. *IEEE Journal on Selected Areas in Communications, Volume 8, Number 3*, pp. 413-427.

123 Little, T., Ahanger, G., Folz, R., Gibbon, J., Reeve, F., Schelleng, D., & Venkatesh, D. (1993). A Digital On-Demand Video Service Supporting Content-Based Queries. *Proceedings of ACM Multimedia '93, California*, pp. 427-436.

124 Little, T. D. C, & Venkatesh, D. (1994). Prospects for Interactive Video-On-Demand. *IEEE Multimedia, Fall 1994*, pp. 14-24.

125 Long, J. (1987). A Framework for User Models. *Proceedings of the Ergonomics Society Annual Conference, London.*

126 Luther, A. C. (1994). Video Technology. In Buford, J. F. K. (Eds.), *Multimedia Systems* (pp. 109-142). ACM Press, New York, New York.

127 Mackay, W. E., & Davenport, G. (1989). Virtual Video Editing in Interactive Multimedia Applications. *Communications of the ACM, Volume 23, Number 7*, pp. 802-810.

128 Mackay, W. E. (1989). EVA: An Experimental Video Annotator for Symbolic Analysis of Video Data. *SIGCHI Bulletin, Volume 21, Number 2*, pp. 68-71.

129 Mackay, W. E., & Pagani, D. S. (1994). Video Mosaic: Laying Out Time in a Physical Space. *Proceedings of ACM Multimedia '94, San Francisco, CA USA*, pp. 165-172.

130 Mackay, W. E., & Beaudouin-Lafon, M. (1998). DIVA: Exploratory Data Analysis with Multimedia Streams. *Proceedings of CHI '98, Los Angeles, CA USA*, pp. 416-423.

131 Macleod, M. (1993). *DRUM: Diagnostic Recorder for Usability Measurement.* NPL Technical Report.

132 Marshal, C. C., Halasz, F. G., Rogers, R. A., & Jensen, W. C. (1991). Aquanet: a Hypertext Tool to Hold Your Knowledge in Place. *Proceedings of Hypertext '91*, pp. 261-275.

133 McLean, P. (1991). *An Introduction to RAID: Redundant Arrays of Inexpensive Discs.* Digital Equipment Corporation. http://www.unix.digital.com/products/raid-paper/index.html.

134 Meng, J., & Chang, S-F. (1996). CVEPS - A Compressed Video Editing and Parsing System. *Proceedings of ACM Multimedia '96, Boston, MA, USA*, pp. 43-53.

135 Metz, C. (1974). *Film Language.* New York: Oxford University Press.

136 Mey, V., & Gibbs, S. (1993). A Multimedia Component Kit, Experiences with Visual Composition of Applications. *Proceedings of ACM Multimedia '93, California*, pp. 291-300.

137 Mills, M., Cohen, J., & Wong, Y. Y. (1992). A Magnifier Tool For Video Data. *Proceedings of ACM CHI '92, Monterey, CA*, pp. 93-98.

138 Minsky, M. L. (1975). A Framework for Representing Knowledge. In Winston, P. K. (Eds.), *The Psychology of Computer Vision* (pp. 211-275). McGraw-Hill.

139 Nack, F., & Parkes, A. (1997). The Application of Video Semantics and Theme Representation in Automated Video Editing. *Multimedia Tools and Applications, Volume 4*, pp. 57-83.

140 Nanard, J., & Nanard, M. (1991). Using Structured Types to Incorporate Knowledge in Hypertext. *Proceedings of Hypertext '91*, pp. 329-343.

141 Nanard, J., & Nanard, M. (1995). Hypertext Design Environments and the Hypertext Design Process. *Communication of the ACM, Volume 38, Number 8*, pp. 49-56.

142 Nardi, B. A., Schwarz, H., Kuchinsky, A., Leichner, R., Whittaker, S., & Sclabassi, R. (1993). Turning Away from Talking Heads: The Use of Video-as-Data in Neurosurgery. *Proceedings of Interchi '93*, pp. 327-334.

143 Nierstrasz, O. (1989). A Survey of Object-Oriented Concepts. In Kim, W., & Lochovsky, F. H. (Eds.), *Object-Oriented Concepts, Databases, and Applications* (pp. 3-21). ACM Press.

144 O' Conaill, B., Whittaker, S., & Wilbur, S. (1993). Conversations Over Video Conferences: An Evaluation of the Spoken Aspects of Video-Mediated Communication. *Human Computer Interaction, Volume 8*, pp. 389-428.

145 O' Day, V. L., & Jeffries, R. (1993). Orienteering in an Information Landscape: How Information Seekers Get From Here to There. *Proceedings of ACM InterCHI '93*, pp. 438-445.

146 Oinas-Kukkonen, H. (1998). What is Inside a Link? *Communications of the ACM, Volume 41, Number 7*, pp. 98.

147 Olson, G. M., Herbsleb, J. D., & Rueter, H. H. (1994). Characterising the Sequential Structure of Interactive Behaviours Through Statistical and Grammatical Techniques. *Human Computer Interactions, Volume 9*, pp. 427-472.

148 Palmer, S. E. (1978). Fundamental Aspects of Cognitive Representation. In Rosch, E., & Lloyd, B. B. (Eds.), *Cognition and categorisation* (pp. 259-303). Hillsdale, N.J., Lawrence Erlbaum.

149 Park, J. (1997). Geographic Information Systems and Problem Solving Environment. *Crossroads, The ACM Student Magazine, Volume 4, Number 1*, pp. 3-8.

150 Parunak, H. V. D. (1989). Hypermedia Topologies and User Navigation. *Proceedings of Hypertext'89*, pp. 43-50.

151 Pass, G., Zabih, R., & Miller, J. (1996). Comparing Images Using Colour Coherence Vectors. *Proceedings of ACM Multimedia '96, Boston, MA, USA*, pp. 65-73.

152 Pearl, A. (1989). Sun's Link Service: A Protocol for Open Linking. *Proceedings of ACM Hypertext '89*, pp. 137-146.

153 Pentland, A., Picard, R. W., Sclaroff, S. (1994). Photobook: Tools for Content-Based Manipulation of Image Databases. *SPIE, Volume 2185*, pp. 34-47.

154 Pentland, A., Picard, R., Davenport, G., & Haase, K. (1994). Video and Image Semantics: Advanced Tools for Telecommunications. *IEEE Multimedia, Summer 1994*, pp. 73-75.

155 Pfeiffer, S., Lienhart, R., Fischer, S., & Effelsberg, W. (1996). Abstracting Digital Movies Automatically. *Journal of Visual Communication and Image Representation, Volume 7, Number 4*, pp. 345-353.

156 Poston, R. M. (1994). Automated Testing from Object Models. *Communications of the ACM, Volume 37, Number 9*, pp. 48-58.

157 Prabhakaran, B., & Raghavan, S. V. (1993). Synchronisation Models for Multimedia Presentation with User Participation. *Proceedings of ACM Multimedia '93, CA, USA*, pp. 157-166.

158 Price, R. (1993). MHEG: An Introduction to the Future International Standard for Hypermedia Object Interchange. *Proceedings of ACM Multimedia '93, CA, USA*, pp. 121-128.

159 Puttress, J. J., & Guimaraes, N. M. (1990). The Toolkit Approach to Hypermedia. *Proceedings of the European Conference on Hypertext '90*, pp. 25-37.

160 Qazi, N. U., Woo, M., & Ghafoor, A. (1993). A Synchronisation and Communication Model for Distributed Multimedia Objects. *Proceedings of ACM Multimedia '93, CA, USA*, pp. 147-155.

161 Reek van, B. (1995). *The Amsterdam Channel*. http://www.channels.nl/adam.html.

162 Rheiner, M. (1992). Object Description and Representation for Visual Multimedia Databases. In Knuth, E., & Wegner, L. M. (Eds.), *Visual Database Systems II* (pp. 331-345), North-Holland.

163 Ritter, F. E., & Larkin, J. H. (1994). Developing Process Models as Summaries of HCI Action Sequences. *Human Computer Interactions, Volume 9*, pp. 345-383.

164 Rosch, E., & Mervis, C. B. (1975). Family Resemblances: Studies in the Internal Structure of Categories. *Cognitive Psychology, Volume 7*, pp. 573-605.

165 Rowe, L., Boreczky, J., & Eads, C. (1994). Indexes for User Access to Large Video Databases. *SPIE Volume 2185*, pp.150-161.

166 Rubin, A., Bresnahan, S., & Ducas, T. (1996). Cartwheeling Through CamMotion. *Communications of the ACM, Volume 39, Number 8*, pp. 84-85.

167 Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorensen, W. (1991). *Object-Oriented Modelling and Design*. Prentice Hall International Inc.

168 Sack, W. & Davis M. (1994). IDIC: Assembling Video Sequences for Story Plans and Content Annotation. *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pp. 30-36.

169 Sanderson, P. M. (1994). *Exploratory Sequential Data Analysis (ESDA) in HCI: Software Support*. Technical report of Engineering Psychology Research Laboratory: EPRL-94-01 (January).

170 Sanderson, P. M., & Fisher, C. (1994). Exploratory Sequential Data Analysis: Foundations. *Human Computer Interaction, Volume 9*, pp. 251-317.

171 Sanderson, P. M., McNeese, M., & Zaff, B. (1994). Knowledge Elicitation and Observation in Engineering Psychology: MacSHAPA and Cogent. *Behaviour Research Methods, Instrument, and Computers, Volume 26, Number 2*, pp. 117-124.

172 Sanderson, P., Scott, J., Johnson, T., Mainzer, J., Watanabe, L., & James, J. (1994). MacSHAPA and the Enterprise of Exploratory Sequential Data Analysis. *International Journal of Human Computer Studies, Volume 41, Number 5*, pp. 633-681.

173 Sawhney, N., Balcom, D., & Smith, I. (1996). HyperCafé: Narrative and Aesthetic Properties of Hypervideo. *Proceedings of ACM Hypertext '96, Washington DC*, pp. 1-10.

174 Sawhney, N., Balcom, D., & Smith, I. (1997). Authoring and Navigating Video in Space and Time. *IEEE Multimedia, Volume 4, Number 4*, pp. 30-39.

175 Schnase, J. L., Leggett, J. J., Hicks, D. L., & Szabo, R. L. (1993). Semantic Data Modelling of Hypermedia Associations. *ACM Transactions on Information Systems, Volume 11, Number 1*, pp. 27-50.

176 Schnorf, P. (1993), Integrating Video into an Application Framework. *Proceedings of Multimedia '93, CA USA*, pp. 411-417.

177 Schütt, H. A., & Straitz, N. A. (1990). HyperBase: A Hypermedia Engine Based on a Relational Database Management System. *Proceedings of the European Conference on Hypertext, Versaille, France*, pp. 95-108.

178 Schwabe, D., & Rossi, G. (1995). The Object-Oriented Hypermedia Design Model. *Communication of the ACM, Volume 38, Number 8*, pp. 45-46.

179 Shahraray, B., & Gibbon, D. C. (1995). Automated Authoring of Hypermedia Documents of Video Programs. *Proceedings of Multimedia '95, CA USA*, pp. 401-409.

180 Sherman, M., Hansen, W. J., McInerny, M., & Neuendorffer, T. (1990). Building Hypertext on a Multimedia Toolkit: An Overview of Andrew Toolkit Hypermedia Facilities. *Proceedings of ECHT '90*, pp. 13-24.

181 Shostak, S. (1991). HDTV: Defining the Future of Broadcasting and Film? *American Cinematographer, Volume 71, Number 8*, pp. 55-60.

182 Simonnet, B., & Smail, M. (1996). Model for Interactive Retrieval of Video and Still Images. *Multimedia Database Systems: Design and Implementation Strategies, Nwosu. K. C., Thuraisingham, B., & Berra, P. B. (eds.)*, pp. 278-297.

183 Smith, M. A., & Christel, M. G. (1995). Automating the Creation of a Digital Video Library. *Proceedings of Multimedia '95, CA USA*, pp. 357-358.

184 Smith, J. R., & Chang, S-F. (1996). VisualSEEk: A Fully Automated Content-Based Image Query System. *Proceedings of ACM Multimedia '96, Boston, MA, USA*, pp. 87-98.

185 Smoliar, S., & Zhang, H. (1994). Content-Based Video Indexing and Retrieval. *IEEE Multimedia, Summer 1994*, pp. 62-72.

186 Swanberg, D., Shu, C-F., & Jain R. (1993). Knowledge Guided Parsing in Video Databases. *SPIE, Volume 1908*, pp. 13-24.

187 Swartz, J., & Smith, B. C. (1995). A Resolution Independent Video Language. *Proceedings of Multimedia '95, CA USA*, pp. 179-188.

188 Takahashi, K. (1998). Metalevel Links: More Power to Your Links. *Communications of the ACM, Volume 41, Number 7*, pp. 103-105.

189 Tanaka, K., Nishikawa, N., Hirayama, S., & Nanba, K. (1991). Query Pairs as Hypertext Links. *Proceedings of the 7th International Conference on Data Engineering, Piscataway, N. J.*, pp. 456-463.

190 Taniguchi, Y., Akutsu, A., Tonomura, Y., & Hamada, H. (1995). An Intuitive and Efficient Access Interface to Real-Time Incoming Video Based on Automatic Indexing. *Proceedings of Multimedia '95, CA USA*, pp. 25-33.

191 Teodosio, L. A., & Bender, W. (1993). Salient Video Stills: Content and Context Preserved. *Proceedings of ACM Multimedia '93, CA, USA*, pp. 39-46.

192 Teodosio, L. A., & Mills, M. (1993). Panoramic Overviews For Navigating Real-World Scenes. *Proceedings of ACM Multimedia '93*, pp. 359-364.

193 Tonomura, Y. (1991). Video Handling Based on Structured Information for Hypermedia Systems. *Proceedings of ACM International Conference on Multimedia Information Systems '91*, pp. 333-344.

194 Tonomura, Y., Akutsu, A., Otsuji, K., & Sadakata, T. (1993). VideoMap and VideoSpaceIcon: Tools for Anatomizing Video Content. *Proceedings of ACM InterCHI '93*, pp. 131-136.

195 Tonomura, T., Akutsu, A., Taniguchi, Y., & Suzuki, G. (1994). Structured Video Computing. *IEEE Multimedia, Fall 1994*, pp. 34-43.

196 Travers, M. (1989). A Visual Representation for Knowledge Structures. *Proceedings of ACM Hypertext '89*, pp. 147-158.

197 Tse, T. H. (1991). *A Unifying Framework for Structured Analysis and Design Models: An Approach using Initial Algebraic Semantics and Category Theory.* Cambridge University Press.

198 Ueda, H., Miyatake, T., Sumino, S., & Nagasaka, A. (1993). Automatic Structure Visualisation for Video Editing. *Proceedings of ACM Multimedia '93, California*, pp. 137-141.

199 Yu, A. & Chen, J. (1995). *The POSTGRES95 User Manual.* Computer Science Division., Dept. of EECS, University of California at Berkley. Available at: http://www.eol.ists.ca/~dunlop/postgres95-manual/.

200 Venema, Y. (1991). A Modal Logical for Chopping Intervals. *Journal of Logic Computation, Volume 1, Number 4,* pp. 453-476.

201 Vernick, M., Venkatramani, C., & Chiueh, T. (1996). Adventures in Building the Stony Brook Video Server. *Proceedings of ACM Multimedia '96, Boston, MA, USA,* pp. 287-295.

202 Vortac, O. U., Edwards, M. B., & Mannning, C. A. (1994). Sequences of Actions for Individual and Teams of Air Traffic Controllers. *Human Computer Interaction, Volume 8, 1993,* pp. 319-343.

203 Wactlar, H. D., Kanade, T., Smith, M. A., & Stevens, S. M. (1996). Intelligent Access to Digital Video: Informedia Project. *IEEE Computer, May 1996,* pp. 46-52.

204 Way, P. (1973). *Your Guide to Cats & Kittens.* Oxley Press Ltd., Great Britain.

205 Wu, J. K., Narasimhalu, A. D., Mehtre, B. M., Lam, C. P., & Gao, Y. J. (1995). CORE: A Content-Based Retrieval Engine for Multimedia Information Systems. *Multimedia Systems, Volume 3, Number 1,* pp. 25-41.

206 Yankelovich, H., Haan, B. J., Meyrowitz, N. K., & Drucker, S. M. (1988). Intermedia: The Concept and Construction of a Seamless Information Environment. *IEEE Computer, Volume 21, Number 1,* pp. 81-96.

207 Yeo, B-L., & Liu, B. (1995). Rapid Scene Analysis on Compressed Video. *IEEE Transactions on Circuits and Systems for Video Technology, Volume 5, Number 6,* pp. 533-543.

208 Yeo, B-L., & Yeung, M. B. (1997). Retrieving and Visualising Video. *Communication of the ACM, Volume 40, Number 12,* pp. 43-52.

209 Yeung, M. M., & Yeo, B-L. (1997). Video Visualisation for Compact Presentation and Fast Browsing of Pictoral Content. *IEEE Transactions on Circuits and Systems for Video Technology, Volume 7, Number 5,* pp. 771-785.

210 Young, E. C. (1979). *The New Penguin Dictionary of Electronics.*Penguin Books.

211 Zabih, R., Miller, Justin, & Mai, K. (1995). A Feature-Based Algorithm for Detecting and Classifying Scene Breaks. *Proceedings of Multimedia '95, CA USA,* pp. 189-200.

212 Zhang, H., Gong, Y., Smoliar, S., & Tang, S. (1994). Automatic Parsing of News Video. *Proceedings of the International Conference on Multimedia Computing and Systems, Los Alamitos, California,* pp. 45-54.

213 Zhang, H. J., Low, C. Y., Smoliar, S. W., & Wu, J. H. (1995). Video Parsing, Retrieval and Browsing: An Integrated and Content-Based Solution. *Proceedings of Multimedia '95, CA USA,* pp. 15-24.

Glossary

Annotation

Some data stored separately, but associated with a segment. Typically an annotation provides some description of some aspect of the content of its associated segment *e.g.* a topic description, or list of characters.

Application

A collection of schemes used for one purpose. It is important to differentiate this from a system which is the hardware and software needed to manipulate the schemes, models, and video.

Base object type

Base object types define the default characteristics of object types. There are two main kinds of object types: units and collections. units are considered atomic objects whereas collections can be composed of other (member) objects. collections themselves have three basic types which determine how their member objects are related: Un-Ordered, Ordered, Temporal.

Base relationship type

Base relationship types define the default name and constraints of relationship types and provide some simple semantics. There are four base relationship types: Membership, Sequential, Temporal, Schema specific.

Celluloid film

A storage medium invented by Kodak in 1889 [21]. Celluloid is used as the storage medium in conventional cinema where it is presented to viewers by projection.

Clip

A sequence of frames regarded as a logical unit. Clips cannot share frames with other clips.

Collection

In VCMF, a Collection is an object which can be composed (using relationships) of other (member) objects.

Context

Context is considered when data items are reused. There are two forms of context:

a) *data* context: a data item's relationship to other data items within an application.

b) *application* context: the application within which a data item is being used.

When data items (*e.g.* objects) are reused their context changes.

Cut

In film making this involves the physical cutting of a shot to an appropriate temporal length for its purpose.

Edit

In the film making process editing refers to the selection and cutting of appropriate takes for shots, the sequencing of these shots into scenes and the sequencing of scenes into a whole film.

Film

a) In this thesis: a particular genre of moving image production typified by the motion pictures produced by studios such as Paramount, Elstree *etc.* Films are stored on some storage medium such as video cassette or celluloid film.

b) Abbreviation of celluloid film.

Frame

a) A still image. A sequence of frames are presented to the viewer in quick succession to create the illusion of a moving image.

b) From Minsky's frames [138]: a structured representation of some entity. Aspects of the entity are represented by slots which are named and can have various values.

Instance

VCMF's description of a video content model. It is instantiated from a schema which defines its structure and semantics. An instance contains objects and relationships between them. Objects represent a model's segments and associated annotations.

Model

a) In this thesis: abbreviation of video content model.
b) In general terms, something which represents some aspects of something else. The reduction in aspects modelled is systematic and explicit, and determined by the purpose of the model.

Moving image

The illusion of a moving image is created by presenting a series of still images to the viewer in quick succession. Typically 25 frames per second are presented to a viewer to create this illusion.

Object

In VCMF an object is descriptive. It describes a segment and its associated annotation in an instance. This is achieved using attributes: temporal attributes to describe the segment, and BNF specified attributes to describe the annotation. An object is an instance of an object type which defines the BNF rules for its attribute values, and determines whether it is a collection or a unit.

Object type

Object types define the possible attribute values of objects in VCMF using BNF rules. They are part of a schema and are themselves defined from base object types.

Play back

To present the frames of a video to a viewer; to play them back to the viewer

Post production

In film making, the editing and dubbing of shots to create the film.

Pre production

In film making, the generation of scripts and storyboards for the film to be produced.

Production

In film making, the filming of takes.

Relationship

In VCMF a relationship can exist between two objects. It can have a normal and reverse name. It is an instance of a relationship type which defines constraints on the relationship.

Relationship type

A relationship type defines constraints and the name of relationships instantiated from it in VCMF. Three kinds of constraint are imposed: constraints on the types of objects the relationship can exist between, constraints on the cardinality of the relationship, and constraints based on objects' attribute values. Relationship types are themselves defined from base relationship types.

Representative frame

A frame which is intended to be representative of the content of a segment, clip, or video document. Also known in other research as: key frame, and Rframe.

Reuse

The use, either by copying or reference, of something in a different context.

Scene

In film making, a scene represents a distinct phase of action occurring within a relatively unified space and time (relative to the film). Scenes are composed of sequences and shots.

Schema

VCMF's description of a scheme. It contains object and relationships types which describe the structural and semantic regularities of instances instantiated from it.

Scheme

Abbreviation of: Video content modelling scheme

Segment

A sequence of frames regarded as a logical unit. Segments may be composed of other segments, and may share frames.

Shot

In film making, a single uninterrupted sequence of frames from one camera.

Studio mode of film making The studio mode of film making was developed in the 1920s by Hollywood studios. It is often likened to the production line assembly of cars as there is a well defined structure to the production process: pre production followed by production followed by post production.

System The software and hardware system used to manipulate video, video content models, and their schemes.

Take In film making a take is an uninterrupted sequence of frames from one camera. The best take for a particular part of the script or storyboard is selected in editing and referred to as a shot

Unit In VCMF a unit is an object which is considered atomic; it cannot be composed of other objects.

Video An electronically stored, transmitted, and displayed temporal medium; a progression of frames suggest a moving image to the viewer [191].

Video content An interpretation of the meaning of the moving image presented to the viewer as the video is played back.

Video content model A model of the content of a video document. That is, it is a representation of an interpretation of the meaning of the image presented to the viewer as the video document is played back.
This thesis considers a video content model to contain segments, associated annotations, and relationships between annotations.

Video content modelling scheme The general structure and semantics of a video content model is defined by a video content modelling scheme (scheme for short). As such, a scheme describes the structural and semantic regularities of a class of video content models.

Video document The largest logical unit of information in a video; it might be the whole video recording itself.

Sections

Programme Background

The Projects

eLib-related events

eLib related Documents

eLib Supporting Studies

Search Page

Programme Contact details

Other Resources

"Information Ecologies" conference details

eLib Mailing lists

Publicity kit

Project Glossaries (forthcoming)

eLib Programme Background

Chris Rusbridge, Director of the UK eLib Programme, wrote an overview of the history and direction of the UK eLib Programme for the July/August issue D-lib magazine, "Towards the Hybrid Library". This is an excellent summary of the whole eLib Programme, and introduces its main themes and concerns, both those behind its creation, and those arising from its development.

John Kirriemuir's earlier "Background to the eLib Programme" is still available.

[Top of Page]

The **Electronic Libraries Programme (eLib)** is funded by the **Joint Information Systems Committee (JISC)**
Page version: 1; Web page maintained by **eLib web-support** and hosted by **UKOLN** - feedback to **elib@ukoln.ac.uk**.
Queries about the programme itself should be directed to the **eLib Programme Office**
Site last revised on: Tuesday, 24-Nov-98 14:21:02 GMT

DC Metadata

Appendix A VCMF Protocol

This appendix details the VCMF protocols used to communicate between toolset components. Figure 62 illustrates a simplified version of the toolset architecture; before each protocol table a small version of this figure is presented which highlights the communication paths and components involved.

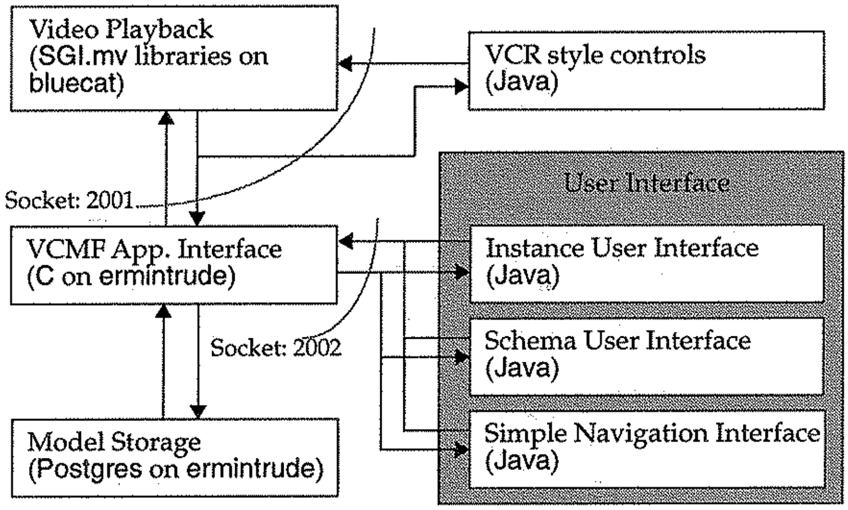
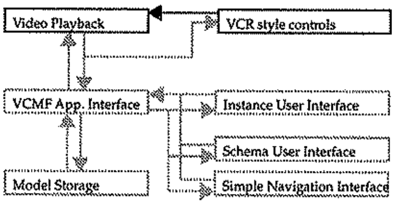


Figure 62: Simplified toolset architecture

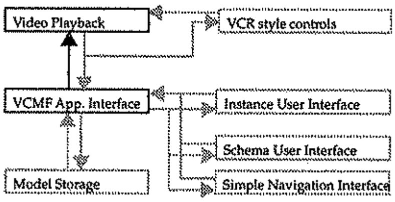
Notes on communication protocols:

- All messages are terminated by a newline. \n in indicates that a newline is to be sent.
- It should be noted that currently message should not contain apostrophes.
- Arrays are sent as postgres arrays *i.e.* {"string", "string",...}
- The VCMF Application Interface always returns a copy of the request followed by a newline character, then the number of fields per line and the number of lines returned, followed by a newline, and then the information requested. The order of the information, and the information returned depends on the mask sent with the request (a copy of this mask is returned in the first line of the reply).
- Mask definitions are given in Appendix B.
- Frame numbers currently refer to SGI.mv library frame numbers.
- The protocols were defined for an earlier version of VCMF and so are no longer obvious abbreviations of their use *e.g.* So is used to retrieve object types as they used to be referred to as schema objects.



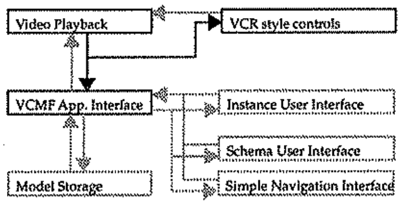
Message	Description
p	Play
s	Stop
t	Restart
m	Toggle audio muting
l	Toggle loop state
l *0/1/2*	Set loop state: *none/ continuous/ swinging*
b	Play backward
f	Play fast
h	Play slow
e	Toggle play every frame
+	Step forward 1 frame
-	Step back one frame
a	Skip to first frame of segment
z	Skip to last frame of segment
g *xxxxxx*	Goto frame *xxxxxx*
r	Play fast backward

Table 22: VCR Style Controls → Video Playback



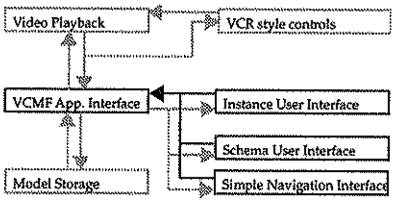
Message	Description
n *xxxxxx yyyyyy*	Move to segment with start_frame *xxxxxx* and end_frame *yyyyyy*

Table 23: VCMF Application Interface → Video Playback



Message	Description
u *xxxxxx*	User caused stop at frame *xxxxxx*
s *xxxxxx*	Stopped at end of segment at frame *xxxxxx*
m *0/1*	Audio mute *on/off*
l *0/1/2*	Loop state *none/loop/swing*
e *0/1*	Play every frame *on/off*

Table 24: Video Playback → VCMF Application Interface & VCR style controls

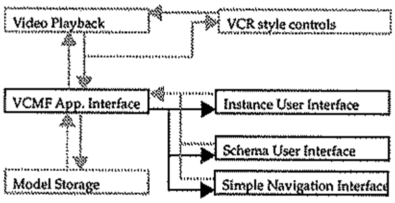


Message	Description
f *xxxx*	Follow relationship id *xxxx*
o *xxxx*	Set current object id to *xxxx*
il *mask*	Get a list of all relationships using the *mask*
io *mask*	Get a list of all objects using the *mask*
sl *mask*	Get a list of all relationship types using the *mask*
so *mask*	Get a list of all object types using the *mask*
sO *mask*	Get list of base object types using *mask*
SO *mask*	Get list of non-base object types using *mask*
co *xxxx mask*	Create object from object type *xxxx, mask* defines what is to be returned after creation
cO *xxxx mask*	Create object type from base object type *xxxx*, return information using *mask*
cl *xxxx yyyy zzzz mask*	Create relationship between object *yyyy* and object zzzz from relationship type *xxxx. mask* defines what is to be returned after creation
CL *mask*	Create new relationship type. Return information using *mask*
cL *xxxx yyyy zzzz mask*	Create relationship type *xxxx* between object types *yyyy* and zzzz. Return newly created object using *mask*
So *xxxx mask*	Get information for object type *xxxx* using *mask*
Sl *xxxx mask*	Get information for relationship type *xxxx* using *mask*
Io *xxxx mask*	Get information for object *xxxx* using *mask*
Il *xxxx mask*	Get information for relationship *xxxx* using *mask*
Mo *xxxx mask*\n *attributes....*	Modify object *xxxx* using *mask* (*attributes* follow in order of *mask*)
Ml *xxxx mask*\n *attributes....*	Modify relationship *xxxx* using *mask* (*attributes* follow in order of *mask*)
ML *xxxx yyyy zzzz mask*\n *info....*	Modify relationship type with TID *xxxx*, from *yyyy*, to *zzzz*. *mask* specifies what is to be returned
mL *xxxx mask*	Modify base relationship type TID *xxxx* using *mask*
u *xxxx*	Set current user id to *xxxx*
Do *xxxx*	Delete object *xxxx*
Dl *xxxx*	Delete relationship *xxxx*

Table 25: User Interface → VCMF Application Interface

Message	Description
DO *xxxx mask*	Delete object type TID *xxxx, mask* is used for the list of object types returned
DL *xxxx yyyy* zzzz	Delete relationship type: *xxxx* is TID, *yyyy* is from TID, zzzz is TID, if zzzz = -1 or *yyyy* = -1 then this is a base relationship type
Fs *xxxx*	Set object *xxxx*'s start frame to current video frame
Fe *xxxx*	Set object *xxxx*'s end frame to current video frame
Fr *xxxx*	Set object *xxxx*'s representative frame to current video frame
Gm	Get list of current messages
Of *xxxx mask relationship_mas*k	Get list of objects related from object *xxxx* - *mask* is for object, *relationship_mask* is optionally for the relationship details
Ot *xxxx mask relationship_mask*	Get list of objects related to object *xxxx* - *mask* is for object, *relationship_mask* is optionally for the relationship details
Lf *xxxx mask*	Get list of relationships from object *xxxx* using *mask*
Lt *xxxx mask*	Get list of relationships to object *xxxx* using *mask*
V *xxxx*	Set the currently displayed video object to object *xxxx*
Rm *xxxx yyyy*	Run instance from object *xxxx*, following relationships with TID *yyyy*
Qs *mask*	Get a list of object types which are appropriate for using as input to Qo (using the *mask*)
Qo *TID mask relationship_mask order_-mask query*	Create a collection by querying the database *TID* is object type of collection to be created, *mask* is mask for return of object, *relationship_mask* is mask for return of relationships created, *order_mask* determines order objects are relationships in, *query* is a psql expression. NOTE - Qo returns two messages
Rs	Stop running the instance

Table 25: User Interface → VCMF Application Interface



Message	Description
Lf *xxxx mask*\n *info....*	List of relationships from object: *xxxx*
Lt *xxxx mask*\n *info....*	List of relationships to object: *xxxx*
Ro *xxxx*\n	Current running object ID is *xxxx*
io *mask*\n *info....*	List of all objects
il *mask*\n *info....*	List of all relationships
so *mask*\n *info....*	List of all object types
SO *mask*\n *info....*	List of all non-base object types
sO *mask*\n *info....*	List of all base object types
sl *mask*\n *info....*	List of all relationship types
co *xxxx mask*\n *info....*	Object created - returning the setup attributes as defined by the *mask*
cl *xxxx mask*\n *info....*	Object created - returning the setup attributes as defined by the *mask*
CL *mask*\n *info....*	Confirmation of creation of relationship type
cL *xxxx yyyy zzzz mask*\n *info....*	Confirmation of creation of relationship type based on base relationship type *xxxx* between object types *yyyy* and *zzzz*
Sl *xxxx mask*\n *info....*	Information about relationship type *xxxx* using *mask*
So *xxxx mask*\n *info....*	Information about object type *xxxx* using *mask*
Io *xxxx mask*\n *info....*	Information about object *xxxx* using *mask*
Il *xxxx mask*\n *info....*	Information about relationship *xxxx* using *mask*
Mo *xxxx mask*\n *info....*	Information about modified object *xxxx* using *mask*

Table 26: VCMF Application Interface → User Interfaces

Message	Description
Ml *xxxx mask*\n *info....*	Information about the modified relationship *xxxx* using *mask*
ML *xxxx yyyy* zzzz *mask*\n *info....*	Confirmation of modification of relationship type uniquely identified by TID *xxxx*, source *yyyy*, destination *zzzz*
mL *xxxx mask*\n *info....*	Confirmation of modification of base relationship type TID *xxxx*
u *xxxx*	Confirmation of new current user's id *xxxx*
Fs *xxxx*\n frame_no	*frame_number* of new start frame of object *xxxx*
Fe *xxxx*\n frame_no	*frame_number* of new end frame of object *xxxx*
Fr *xxxx*\n frame_no	*frame_number* of new representative frame of object *xxxx*
Do *xxxx* *yyyy...*	Confirmation of deletion of object *xxxx*, *yyyy...* is \n separated list of other objects and relationships deleted
DO *xxxx mask*\n *info...*	Confirmation of deletion of object type TID *xxxx*, returning list of object types deleted
Dl *xxxx*	Confirmation of deletion of relationship *xxxx*
DL *xxxx yyyy* zzzz	Confirmation of deletion of relationship type uniquely identified by TID *xxxx*, source *yyyy*, destination *zzzz* ($x = y = -1$ if it is a base relationship type)
Of *xxxx mask relationship_mask*\n *info...*	List of objects and relationships from object *xxxx*
Ot *xxxx mask relationship_mask*\n *relationships...*	List of objects and relationships to object *xxxx*
Gm\n *info...*	List of current messages (\n separated)
V *xxxx*	Confirmation of setting currently video displayed object to *xxxx*
Qs *mask*	List of object types which are appropriate for input to Qo (using *mask*)
Qo *schema_ID mask relationship_mask order_attribute query* collection object created Qo *schema_ID mask relationship_mask order_attribute query* relationship objects created	TWO messages returned by Qo - first the collection object that was created then the relationships that were created
Rm *xxxx yyyy*	Confirmation of started running the instance from object *xxxx* following relationships with TID = *yyyy*
Rs *xxxx*	Instance has stopped running at object *xxxx* either: a) in response to Rs b) came to end of relationships

Table 26: VCMF Application Interface → User Interfaces

Appendix B Databases Tables and Fields

This appendix details the use of table rows in the Postgres database. Each table in this appendix presents information about one database table. The columns of the tables are used as follows:

- The Use column relates to the VCMF information stored.
- The Information type column details the type of information stored in the database. Information enclosed in [] indicates that it is optional.
- The Database field name column directly relates to the field of the database in which data is stored.
- The Communication mask column defines the mask used to communicate such information using the communication protocol detailed in Appendix A.

  The letters in this column are also used to define which attribute an Attribute-Ordered collection is ordered on (the Ordering attribute of the schema_objects table).

  In addition the following are defined for non Attribute-Ordered collections:

  u: user ordered.

  0: un-ordered.

It is worth noting that some of the field names and terminology is derived from older versions of VCMF which had different terminology.

Table 27 presents the rows for the database table schema_objects which stores information about the object types in the VCMF system.

Use	Information type	Database field name	Communication mask
Type ID	Number	SID	s
Name	String	Name	n
Base object type name	String	Kind	k
Author ID	Number	Author	a
Temporal Extent needed	Boolean	Temporal	t
Children allowed	Type ID of relationship type for membership, or -1 if no children allowed	Indexed_children	c
Ordered children allowed	Type ID of relationship type for sequential, or -1 if no ordered children allowed	Ordered_children	o
Attribute to order Attribute-Ordered collection on	Communication mask of another attribute	Ordering	r
Unique attribute names	Array of strings	App_info[] (text)	f
Necessity of unique attributes	Array of booleans	App_fields[] (integer)	

Table 27: Database rows for table schema_objects

Table 28 defines the information stored about objects by the VCMF system in the instance_objects table. The object type from which an object is defined provides the attribute names for the unique attributes (*e.g.* keywords *etc.*), and determines whether or not the attribute values must be filled, as well as the temporal extent and whether the number of children should be filled in.

Use	Information type	Database field name	Communication mask
Type ID	Number	SID	s
ID	Number	ID	i
Name	String	Name	n
Start frame	[Frame number]	Start_frame	x
End frame	[Frame number]	End_frame	y
Start frame & End frame	[Frame number], [Frame number]	Start_frame, End_frame	t
Number of Children	[Number]	Children	d
Representative info	Frame number	Representative	r
Creator ID	Number	Creator	c
Unique attribute values Specific unique attribute value	Array of strings	App_info[] App_info[A-letter]	a capital letter

Table 28: Database rows for table instance_objects

Table 29 defines the information stored about relationship types by the VCMF system in the schema_links table. Note that defining object types implicitly creates relationship types for object types which are permitted children.

Use	Information type	DB field name	Communication mask
Type ID	Number	SID	s
Name	String	Name	n
Reverse name	[String] (if empty then there is no reverse name)	Reverse_name	r
Cardinality specification	List of numbers or m	Navigational	v
Source constraints	Currently undefined	Source_def	s
Destination constraints	Currently undefined	Dest_def	d
Author ID	Number	Author	a

Table 29: Database rows for table schema_links

Table 30 defines the information stored about relationships by the VCMF system in the instance_links table.

Use	Information type	DB field name	Communication mask
Type ID	Number	SID	s
ID	Number	ID	i
Source ID	Number	Source	f
Destination ID	Number	Destination	t
Creator	Number	Creator	c
Group ID (used by VCMF to keep track of members of collections)	Number	GroupID	g

Table 30: Db rows for table instance_links

Appendix C Database Data

For the sake of completeness this appendix lists the contents of the database tables.

SID	ID	Name	Start_frame	End_frame	Children	Represent ative	Creator	App_info
46	59	Shots of Xanadu	1449	2212	0	1601	1	{"2.a.A"}
46	60	Funeral:headlinesofKanesdeath	2212	2531	0	2220	1	{"2.a.B"}
46	61	Growth of financial empire	2531	2893	0	2531	1	{"2.a.C"}
46	62	Silver mine and Mrs. Kanes boarding house	2893	3020	0	2893	1	{"2.a.D"}
46	63	Thatcher testimony at congressional committee	3020	3473	0	3030	1	{"2.a.E"}
46	64	Political career	3473	4034	0	3799	1	{"2.a.F"}
46	65	Privatelife;weddings,divorces	4034	4314	0	4047	1	{"2.a.G"}
46	66	Opera house and Xanadu	4314	4463	0	4459	1	{"2.a.H"}
46	67	Political campaign	4463	4814	0	4473	1	{"2.a.I"}
46	68	Depression	4814	4950	0	4822	1	{"2.a.J"}
46	69	1935: Kanes old age	4950	5476	0	4961	1	{"2.a.K"}
46	71	Death announced	5654	5742	0	5696	1	{"2.a.M"}
45	56	Reporters discuss "Rosebud"	5742	6607	0	6008	1	{"2.b",""}
45	55	News on the March	1451	5742	13	1476	1	{"2.a",""}
45	106	Thompson enters and reads Thatchers manuscript	7953	8720	0	8565	1	{"4.a",""}
45	108	Kane grows up and buys the Inquirer	10630	11134	0	11102	1	{"4.c","1895"}
45	135	Thompson visits Bernstein	13925	15181	1	13941	1	{"5.a",""}
45	109	Kane launches the Inquierers attack on big business	11134	12536	0	11259	1	{"4.d","1897"}
45	110	The Depression: Kane sells Thatcher his newspaper chain	12536	13682	0	12568	1	{"4.e","1929"}
44	123	First flashback	8720	13682	4	13644	1	{"Flashback"}
45	111	Thompson leaves library	13644	13925	0	13769	1	{"4.f",""}
45	139	Montage: The Inquirers growth	18265	18578	0	18280	1	{"5.c","1900"}
45	155	Thompson talks with Leland	22725	23885	1	23156	1	{"6.a",""}
45	160	Breakfast table montage: Kanes marriage deteriorates	23885	24893	1	24761	1	{"6.b","1915"}
45	141	Leland and Bernstein discuss Kanes trip abroad	20790	21297	0	21063	1	{"5.e","1909"}
44	136	Second flashback	15181	22040	5	15201	1	{"Flashback"}
45	137	Bernstein concludes his reminiscence	22040	22725	0	22054	1	{"5.g",""}
43	134	Bernsteins office	13925	22725	3	13941	1	{"5"}
44	156	Third flashback	23885	24893	1	24893	1	{"Flashback"}
45	157	Leland continues his recollections	24893	25368	0	24973	1	{"6.c",""}
45	161	Kane meets Susan and goes to her room	25368	27894	0	25618	1	{"6.d","1919"}
45	162	Kanes political campaign culminates in his speech	27894	29458	0	28034	1	{"6.e","1920"}
45	193	Thompson talks with Susan	39214	39865	1	39410	1	{"7.a",""}
45	163	Kane confronts Gettys, Emily, Susan	29458	32220	1	30021	1	{"6.f","1920"}
45	164	Kane loses election and Leland asks to be transferred	32220	34704	0	32276	1	{"6.g","1921"}
45	165	Kane marries Susan	34704	35010	0	34723	1	{"6.h","1922"}
45	166	Susans opera premiere	35010	35662	0	35034	1	{"6.i","1923"}
45	167	Because Leland is drunk, Kane finishes Lelands review	35662	38164	0	36493	1	{"6.j","1923"}
44	158	Third flashback (continued)	25368	38164	7	25618	1	{"Flashback"}
45	159	Lelandconcludeshisreminiscence	38164	39214	0	38269	1	{"6.k",""}
43	154	Nursing home	22725	39214	5	23156	1	{"6"}
45	195	Susan rehearses her singing	39865	41003	0	40048	1	{"7.b","1922"}
45	196	Susans opera premier	41003	42624	0	41025	1	{"7.c","1923"}
45	197	Kane insists that Susan goes on singin	42624	43570	0	42694	1	{"7.d","1923"}
45	198	Montage: Susans opera career	43570	43895	0	43730	1	{"7.e","1924"}
45	199	Susan attempt suicide and Kane promises she can quit singing	43895	45228	0	44222	1	{"7.f","1925"}
45	200	Xanadu: Susan bored	45228	46143	0	45330	1	{"7.g","1926"}
45	201	Montage: Susanplays with jigsaw puzzle	46143	46297	0	46154	1	{"7.h","1926"}
45	202	Xanadu: Kane promises picnic	46297	46754	0	46693	1	{"7.i","1927"}
45	203	Picnic: Kane slaps Susan	46754	47748	0	47163	1	{"7.j","1927"}
45	204	Xanadu: Susan leaves Kane	47748	48885	0	48056	1	{"7.k","1928"}
44	194	Fourth flashback	39865	48885	10	48821	1	{"Flashback"}
43	192	El Rancho nightclub	39214	49296	3	39273	1	{"7"}
45	205	Susanconcludesherreminiscence	48885	49296	0	49210	1	{"7.l",""}
45	235	Thompson talks with Raymond	49296	49719	0	49331	1	{"8.a",""}
43	100	El Rancho nightclub: Thompson tries to interview Susan	6607	7953	1	6679	1	{"3"}
43	250	End	53902	54974	1	54904	1	{"E"}

Table 31: Table instance_objects (objects)

SID	ID	Name	Start_frame	End_frame	Children	Representative	Creator	App_Info
45	240	Raymond concludes his reminiscence,Thompsontalkswith the other reporters, all leave	51262	52798	1	51284	1	{"8.c",""}
45	242	SurveyofKanespossessionsleads to a revalation of Rosebud. Exterior of gate and of castle. The end	52798	53902	0	53444	1	
{"8.d",""}								
43	234	Xanadu	49296	53902	4	49315	1	{"8"}
42	48	Citizen Kane	0	54974	10	179	1	{"Voted Greatest Film of All Time by the British Film Institute"}
43	51	Projection room	1451	6607	2	6424	1	{"2"}
43	103	Thatcher library	7953	13925	3	7969	1	{"4"}
43	420	E	65393	65681	0	65650	1	{"E"}
45	430	Titles	55687	55867	0	55812	1	{"C.c",""}
45	444	Kane dies	1006	1372	0	1166	1	{"1.a",""}
43	417	Credit Title	54974	55867	3	55812	1	{"C"}
45	438	Welles - a genius	57008	57157	0	57023	1	{"1.c",""}
43	418	Orson Welles - the man	55867	57157	3	56702	1	{"1"}
43	50	Xanadu: Kane dies	249	1449	1	1178	1	{"1"}
45	428	Introduction to film and controversy	54974	55317	1	55110	1	{"C.a",""}
47	447	Interview: Greatest film of all time	55317	55405	0	55332	1	{"Robert Wise"}
45	107	Kanes mother sends the boy off with Thatcher	8720	10630	2	9022	1	{"4.b","1871"}
47	449	Welcome home Mr.Kane	21297	21744	0	0	1	{"Kane, Bernstein and New York Enquirer staff"}
45	142	Kane returns with is fiancee Emily	21297	22040	1	21433	1	{"5.f","1910"}
47	453	Kane walks through Xanadu alone	51102	51262	0	51181	1	{"Old Kane"}
45	238	Kane destroys Susans room and picks up paperweight murmering Rosebud	49719	51262	1	50646	1	{"8.b","1928"}
45	429	Greatest movie of all time	55317	55687	4	55658	1	{"C.b",""}
45	463	Welles idea for Citizen Kane and work with Mankowitz	56702	57824	0	56702	1	{"2.a",""}
45	464	Writing of screenplay with Mankowitz - breaking usual Hollywood styles	57824	58615	0	57838	1	{"2.b",""}
45	465	Overview of cast	58615	59168	0	58622	1	{"2.c",""}
45	466	Problem with RKO over Citizen Kane	59168	59551	0	59207	1	{"2.d",""}
45	468	Use of angles and spatial relationships	59551	60506	0	60248	1	{"2.e",""}
45	469	Use of deep focus	60506	60881	0	60604	1	{"2.f",""}
45	470	Lighting techniques	60881	61461	0	60907	1	{"2.g",""}
45	471	Use of models	61461	61696	0	61544	1	{"2.h",""}
45	472	Soundtrack	61696	62279	0	62271	1	{"2.i",""}
45	473	Film aging techniques	62279	62817	0	62315	1	{"2.j",""}
45	474	Make up techniques	62817	63117	0	62902	1	{"2.k",""}
45	475	Homour in Citizen Kane	63117	63330	0	63136	1	{"2.l",""}
45	476	Musical Score	63330	63603	0	63331	1	{"2.m",""}
45	477	Controversy about Citizen Kane	63603	64582	0	63640	1	{"2.n",""}
43	478	Welles other films	64582	64707	0	64585	1	{"3"}
43	479	Praise of Welles	64707	65393	0	64725	1	{"4"}
42	416	50th Anniversary Documentary	54974	65681	6	55833	1	{"Lots of famous people like it"}
43	419	The Making of Citizen Kane	56702	65393	8	57176	1	{"2"}
43	49	Credit title	0	249	1	179	1	{"C"}
44	467	Techniques used in Citien Kane	59551	63117	7	59558	1	{"Other"}
46	522	Welles and Mercury Credits	99	158	0	157	1	{"C.a.A"}
47	451	For the union flag	8788	9598	0	8854	1	{"Boy Charles, Mother, Father, Thatcher"}
46	514	Welles life montage	55867	56056	0	55877	1	{"1.a.A"}
47	515	Discussion of War of the Worlds	56056	56185	0	56084	1	{"Robert Wise"}
47	516	Creative Welles	56317	56536	0	56329	1	{"Ruth Warwick"}
47	526	Kane photgraphs new staff	18578	18822	0	18717	1	{"Kane and chronicalestaff"}
45	436	BrillianceofWelles-lifestory and transition from radio	55867	56536	5	55932	1	{"1.a",""}

Table 31: Table instance_objects (objects)

SID	ID	Name	Start_frame	End_frame	Children	Representative	Creator	App_Info
47	530	Reasons for Welles success	56626	56697	0	56642	1	{"Ridley Scott"}
46	532	RKO photo montage	56697	56749	0	56707	1	{"1.b.A"}
46	535	Creative freedom photo montage	56857	57008	0	56993	1	{"1.b.B"}
46	538	Kanes success song	19709	20789	0	20586	1	{"5.d.?"}
45	140	Party: the Inquirer celebrates getting the Chronical staff	18578	20790	2	19451	1	{"5.d","1906"}
47	543	Kane enters enquirer	15420	15519	0	15474	1	{"Kane and Zebediah"}
45	437	Welles move to Hollywood in 1939	56536	57008	5	56897	1	{"1.b",""}
549	618	North East Side	0	0	2	854	1	{}
549	580	North Side	0	0	2	576	1	{}
549	574	North West Side	0	0	2	220	1	{}
551	559	North West Corner	0	0	4	0	1	{}
551	586	North East Corner	0	0	4	0	1	{}
551	602	West Gate	0	0	4	0	1	{}
551	624	East Gate	0	0	4	0	1	{}
551	657	South East Corner	0	0	4	0	1	{}
551	682	South East Corner	0	0	4	0	1	{}
549	702	South East Side	0	0	2	0	1	{}
549	680	South Side	0	0	2	2486	1	{}
549	656	South East Side	0	0	2	0	1	{}
549	654	Centre	0	0	2	0	1	{}
551	733	East Road	0	0	4	0	1	{}
549	731	East Road	0	0	2	0	1	{}
548	558	Queens Square	0	0	15	1234	1	{}
550	575	N	66115	66441	0	66115	1	{}
550	640	N	67160	67470	0	67160	1	{}
550	644	W	67481	67634	0	67481	1	{}
550	581	E	66504	66677	0	66504	1	{}
550	620	S	66773	67097	0	66773	1	{}
550	648	S	67644	67958	0	67644	1	{}
550	653	S	68145	68448	0	68145	1	{}
550	681	W	68500	68645	0	68500	1	{}
550	703	N	68738	69045	0	68738	1	{}
550	704	S	69066	69379	0	69066	1	{}
550	714	E	69397	69537	0	69397	1	{}
550	718	N	69550	69848	0	69550	1	{}
550	722	W	69973	70131	0	69973	1	{}
550	723	E	70173	70326	0	70173	1	{}
550	732	E	70653	70691	0	70653	1	{}
550	738	W	70739	70773	0	70739	1	{}
757	758	Tour of Queens Square	0	0	7	0	1	{}
552	560	NtoE	66443	66453	0	66453	1	{}
552	597	NtoE	66720	66727	0	66727	1	{}
552	605	NtoE	66071	66084	0	66084	1	{}
552	629	NtoE	67117	67127	0	67127	1	{}
552	666	NtoE	68467	68477	0	68477	1	{}
552	684	NtoE	68659	68669	0	68669	1	{}
552	737	NtoE	70716	70724	0	70724	1	{}
552	561	EtoS	66453	66465	0	66465	1	{}
552	589	EtoS	66677	66694	0	66694	1	{}
552	607	EtoS	66084	66094	0	66094	1	{}
552	630	EtoS	67127	67135	0	67135	1	{}
552	667	EtoS	68477	68489	0	68489	1	{}
552	685	EtoS	68669	68680	0	68680	1	{}
552	734	EtoS	70691	70699	0	70699	1	{}
552	562	StoW	66465	66478	0	66478	1	{}
552	591	StoW	66694	66706	0	66706	1	{}
552	610	StoW	66094	66104	0	66104	1	{}
552	627	StoW	67097	67107	0	67107	1	{}
552	664	StoW	68448	68458	0	68458	1	{}
552	686	StoW	68680	68690	0	68690	1	{}
552	735	StoW	70699	70708	0	70708	1	{}
552	563	WtoN	66478	66492	0	66492	1	{}
552	594	WtoN	66706	66720	0	66720	1	{}
552	613	WtoN	66104	66116	0	66116	1	{}
552	628	WtoN	67107	67117	0	67117	1	{}
552	665	WtoN	68458	68467	0	68467	1	{}
552	683	WtoN	68645	68659	0	68659	1	{}
552	736	WtoN	70708	70716	0	70716	1	{}
780	804	Billboard of Susan	6607	6645	0	6621	1	{}
780	823	Carter exiting Enquirer building	17126	17168	0	17148	1	{}
780	833	Cobbled street	25368	25393	0	25379	1	{}
780	843	Susan singing	35383	35441	0	35411	1	{}
780	853	Picture of Susan	39214	39236	0	39235	1	{}
780	863	Distant view of Xanadu	45228	45260	0	45248	1	{}
780	873	Singer	46865	46903	0	46873	1	{}

Table 31: Table instance_objects (objects)

SID	ID	Name	Start_frame	End_frame	Children	Representative	Creator	App_info
780	883	Crates in Xanadus cellar	52798	52860	0	52798	1	{}
780	900	No Trespassing sign	249	413	0	293	1	{}
780	929	Thompson gets manuscript	8222	8545	0	8290	1	{}
781	844	Through to top of set	35441	35597	0	35499	1	{}
781	864	Through Xanadu grounds	45260	45288	0	45275	1	{}
781	874	Singing,guestsdancing,barbeque	46903	47103	0	46970	1	{}
781	919	Through sign and skylight	6678	6876	0	6678	1	{}
781	805	Over billboard	6645	6678	0	6651	1	{}
781	902	Over Fence	413	640	0	0	1	{}
781	904	Grounds of Xanadu	640	997	0	934	1	{}
781	931	Over Thompsons shoulder	8545	8598	0	8578	1	{}
781	933	Through manuscript	8598	8721	0	8621	1	{}
782	806	Susan lying on table	6876	6913	0	6873	1	{}
782	825	Kane writing on paper	17318	17363	0	17352	1	{}
782	835	Kane talks to Susan	25585	25646	0	25610	1	{}
782	845	Stage hands disgust	35597	35664	0	35630	1	{}
782	855	Susan and Thompson at table	39353	39385	0	39385	1	{}
782	875	Kane and Susan sit alone	47103	47136	0	47111	1	{}
782	885	Sledge in furnace	53363	53542	0	53446	1	{}
782	906	Kane outlined against window	997	1082	0	1020	1	{}
782	935	Young Kane	8721	8769	0	8765	1	{}
779	842	Susans opera premier farce	35383	35664	3	0	1	{"Ingress",""}
779	862	Susan bored in Xanadu	45228	45331	3	0	1	{"Ingress",""}
779	872	Kane and Susan have a picnic	46865	47136	3	0	1	{"Ingress",""}
967	1001		249	264	0	264	1	{"In"}
978	1002		282	416	0	390	1	{"Up"}
968	1003		398	425	0	425	1	{}
978	1004		416	459	0	438	1	{"Up"}
968	1005		445	463	0	463	1	{}
978	1006		450	528	0	486	1	{"Up"}
968	1007		492	538	0	538	1	{}
968	1008		554	593	0	593	1	{}
968	1009		610	664	0	664	1	{}
968	1010		687	737	0	737	1	{}
968	1011		745	876	0	876	1	{}
968	1012		881	926	0	926	1	{}
968	1013		933	957	0	957	1	{}
968	1014		999	1040	0	1040	1	{}
968	1015		1066	1095	0	1095	1	{}
977	1017		0	0	3	0	1	{}
971	1018		0	0	1	0	1	{}
966	1016		0	0	12	0	1	{}
978	1039		6625	6690	0	6690	1	{"Up"}
979	1040		6686	6791	0	6791	1	{"In"}
978	1041		6751	6791	0	6791	1	{"Down"}
976	1042		6748	6791	0	6791	1	{"Down"}
968	1043		6783	6806	0	6806	1	{}
977	1047		0	0	3	0	1	{}
974	1046		0	0	1	0	1	{}
971	1045		0	0	2	0	1	{}
966	1044		0	0	1	0	1	{}
979	1059		8222	8289	0	8289	1	{"In"}
968	1060		8544	8557	0	8557	1	{}
979	1061		8544	8599	0	8579	1	{"In"}
976	1062		8573	8599	0	8582	1	{"Down"}
968	1063		8592	8604	0	8604	1	{}
973	1064		8592	8617	0	8617	1	{"In"}
970	1065		8621	8622	0	8622	1	{}
975	1066		8622	8727	0	8681	1	{"Right"}
968	1067		8698	8727	0	8727	1	{}
972	1070		0	0	1	0	1	{}
974	1071		0	0	2	0	1	{}
977	1072		0	0	2	0	1	{}
966	1068		0	0	4	0	1	{}
971	1069		0	0	3	0	1	{}
781	824	Newsboy, outisde of building	17168	17249	0	17291	1	{}
781	1089	Through window	17249	17318	0	17276	1	{}
779	822	Kanes first edition completed	17126	17363	4	0	1	{"Ingress",""}
979	1092		0	17200	0	17200	1	{"Out"}
970	1093		17200	17201	0	17201	1	{}
979	1094		17201	17241	0	17224	1	{"In"}
968	1095		17227	17251	0	17251	1	{}
979	1096		17230	17322	0	17302	1	{"In"}
968	1097		17309	17325	0	17325	1	{}
977	1100		0	0	3	0	1	{}
971	1099		0	0	1	0	1	{}
966	1098		0	0	3	0	1	{}

Table 31: Table instance_objects (objects)

SID	ID	Name	Start_frame	End_frame	Children	Representative	Creator	App_info
781	834	Drugstore	25393	25464	0	25464	1	{}
781	1112	Susan	25464	25511	0	25496	1	{}
781	1113	Kane splashed by carriage	25511	25585	0	25549	1	{}
779	832	Kane meets Susan	25368	25646	5	0	1	{"Ingress",""}
975	1118		25368	25399	0	25399	1	{"Left"}
976	1119		25368	25399	0	25399	1	{"Up"}
979	1120		25399	25464	0	25464	1	{"In"}
975	1121		25435	25489	0	25489	1	{"Right"}
975	1122		25497	25531	0	25531	1	{"Right"}
979	1123		25531	25582	0	25582	1	{"In"}
974	1124		0	0	4	0	1	{}
977	1125		0	0	2	0	1	{}
971	1132		0	0	2	0	1	{}
978	1137		35612	0	0	35612	1	{"Up"}
977	1138		0	0	1	0	1	{}
971	1139		0	0	1	0	1	{}
781	1144	Through skylight	39324	39353	0	39353	1	{}
781	854	Billboard	39236	39324	0	39271	1	{}
779	852	Thompsons second visit to Susan	39214	39385	4	0	1	{"Ingress",""}
967	1148		39214	39232	0	39232	1	{"In"}
978	1150		39232	39272	0	39272	1	{"Up"}
979	1151		39272	39357	0	39317	1	{"In"}
976	1152		39328	39357	0	39335	1	{"Down"}
968	1153		39344	39357	0	39357	1	{}
966	1147		0	0	2	0	1	{}
977	1156		0	0	2	0	1	{}
974	1157		0	0	1	0	1	{}
971	1149		0	0	2	0	1	{}
967	1167		45228	45246	0	45246	1	{"In"}
968	1168		45253	45272	0	45272	1	{}
968	1169		45272	45297	0	45297	1	{}
782	865	Susan bored with jigsaw	45288	45306	0	45306	1	{}
966	1170		0	0	3	0	1	{}
975	1176		46894	47041	0	47041	1	{"Left"}
973	1177		0	46955	0	46955	1	{"Out"}
979	1178		47047	47103	0	47101	1	{"In"}
968	1179		47101	47104	0	47104	1	{}
974	1182		0	0	1	0	1	{}
972	1180		0	0	1	0	1	{}
977	1183		0	0	1	0	1	{}
971	1181		0	0	3	0	1	{}
966	1184		0	0	1	0	1	{}
968	1196		52849	52862	0	52862	1	{}
978	1198		52862	53252	0	53252	1	{"Down"}
979	1197		52862	53252	0	53252	1	{"In"}
975	1199		53191	53252	0	53252	1	{"Right"}
970	1200		53253	53254	0	53254	1	{}
979	1201		53254	53330	0	53330	1	{"In"}
975	1202		53254	53330	0	53330	1	{"Right"}
968	1203		53359	53370	0	53370	1	{}
973	1204		53370	53537	0	53537	1	{"In"}
977	1208		0	0	3	0	1	{}
974	1209		0	0	2	0	1	{}
972	1207		0	0	1	0	1	{}
971	1206		0	0	3	0	1	{}
966	1205		0	0	3	0	1	{}
781	884	Over crates	52860	53245	0	52914	1	{}
781	1226	Man moves to furnace	53245	53363	0	53267	1	{}
779	882	Burning of Rosebud sledge	52798	53542	4	0	1	{"Ingress",""}
780	1230	Kanes declaration in Inquirer	18265	18304	0	18274	1	{}
781	1231	Piles of Inquirers	18304	18381	0	18350	1	{}
782	1232	Inquirer window with 26000 circulation	18381	18434	0	18381	1	{}
779	1229	Start of new Inqiurer	18265	18434	3	0	1	{"Recession",""}
978	1241		18276	18384	0	18353	1	{"Up"}
977	1240		0	0	1	0	1	{}
971	1238		0	0	1	0	1	{}
968	1244		18365	18385	0	18385	1	{}
966	1239		0	0	1	0	1	{}
780	1252	Kane at table	24807	24826	0	24807	1	{}
781	1253	Length of table	24826	24854	0	24854	1	{}
782	1254	Kane and wife distanced	24854	24885	0	24885	1	{}
779	1251	Realisationofmarriageproblems	24807	24885	3	24826	1	{"Recession",""}
975	1263		24816	24866	0	24866	1	{"Right"}
979	1262		24816	24877	0	24877	1	{"Out"}
974	1264		0	0	1	0	1	{}
977	1261		0	0	1	0	1	{}

Table 31: Table instance_objects (objects)

SID	ID	Name	Start_frame	End_frame	Children	Represent ative	Creator	App_info
971	1260		0	0	2	0	1	{}
780	1273	GettysandEmilyoutsidelovenest	32073	32166	0	32138	1	{}
781	1274	Outside of lovenest	32166	32252	0	32252	1	{}
782	1275	Headlines about affair	32252	32292	0	32292	1	{}
779	1272	Kanes affair discovered	32073	32292	3	0	1	{"Recession",""}
979	1284		32168	32196	0	32196	1	{"Out"}
968	1285		32209	32233	0	32233	1	{}
973	1286		32240	32279	0	32279	1	{"Out"}
977	1283		0	0	1	0	1	{}
972	1287		0	0	1	0	1	{}
971	1281		0	0	2	0	1	{}
966	1282		0	0	1	0	1	{}
780	1299	Susan and Thompson	49196	49215	0	49215	1	{}
781	1300	Through Skylight	49215	49283	0	49283	1	{}
782	1301	El Rancho nightclub sign	49283	49294	0	49294	1	{}
779	1298	Thompson leaves Susan	49196	49294	3	0	1	{"Recession",""}
976	1313		49213	49267	0	49267	1	{"Down"}
976	1314		49267	49276	0	49276	1	{"Up"}
978	1312		49213	49276	0	49276	1	{"Up"}
979	1311		49213	49294	0	49294	1	{"Out"}
974	1309		0	0	2	0	1	{}
977	1310		0	0	2	0	1	{}
971	1308		0	0	2	0	1	{}
782	1326	Hundreds of boxes	52757	52798	0	52798	1	{}
781	1325	Piles of objects	52485	52757	0	52757	1	{}
780	1324	Thompson with jigsaw	52377	52485	0	52485	1	{}
779	1323	Final overview of Kanes posessions	52377	52798	3	0	1	{"Recession",""}
979	1338		52360	52756	0	52756	1	{"Out"}
978	1339		52474	52756	0	52756	1	{"Up"}
975	1337		52560	52739	0	52739	1	{"Right"}
968	1341		52742	52757	0	52757	1	{}
968	1340		52789	52810	0	52810	1	{}
966	1334		0	0	2	0	1	{}
974	1335		0	0	1	0	1	{}
977	1336		0	0	2	0	1	{}
971	1333		0	0	2	0	1	{}
780	1354	Rosebud sledge in furnace	53466	53552	0	53466	1	{}
781	1355	Smoke from chimney	53552	53688	0	53580	1	{}
781	1356	No tresspassing sign	53688	53804	0	53750	1	{}
782	1357	Gates of Xanadu	53804	53829	0	53829	1	{}
779	1353	Last view of Xanadu	53466	53829	4	0	1	{"Recssion",""}
774	783	Citizen Kane	0	0	16	0	1	{}
973	1369		53466	53530	0	53466	1	{"In"}
976	1371		53530	53688	0	53635	1	{"Up"}
978	1373		53675	53773	0	53773	1	{"Down"}
967	1374		53512	53532	0	53532	1	{"Out"}
967	1375		53532	53563	0	53563	1	{"In"}
968	1376		53677	53717	0	53686	1	{}
972	1368		0	0	1	0	1	{}
974	1370		0	0	1	0	1	{}
977	1372		0	0	1	0	1	{}
971	1366		0	0	3	0	1	{}
966	1367		0	0	3	0	1	{}
980	992	Citizen Kane	0	0	27	0	1	{}
779	897	First view of Kane ~ on death bed	249	1082	4	0	1	{"Ingress",""}
779	918	First view of Susan	6607	6913	4	0	1	{"Ingress",""}
779	936	First flashback about Kane	8222	8769	4	0	1	{"Ingress",""}
45	1390	Credit title	0	249	1	179	1	{"C.a",""}
45	1393	El Rancho nightclub: Thompson tries to interview Susan	6607	7953	0	6679	1	{"3.a",""}
43	1398	Credits	0	0	0	0	1	{"C"}
43	1399	Introduction	0	0	0	0	1	{"1"}
43	1403	Summary	0	0	0	0	1	{"5"}
43	1404	End credits	0	0	0	0	1	{"E"}
42	1397	Flowers today	0	0	6	0	1	{"Documentary about flowers in London"}
45	1419	At home in Bloomsbury	0	0	0	0	1	{"3.a",""}
45	1420	Flowers at QMW	0	0	0	0	1	{"2.a",""}
43	1400	Flowers at work	0	0	1	0	1	{"2"}
43	1401	Flowers at home	0	0	1	0	1	{"3"}
43	1402	Flowers at leisure	0	0	2	0	1	{"4"}
45	1427	Russell square	0	0	0	0	1	{"4.b",""}
45	1429	Hyde park	0	0	0	0	1	{"4.d",""}
45	1428	Regents park	0	0	0	0	1	{"4.c",""}
44	1417	Flowers in squares	0	0	2	0	1	{"Other"}

Table 31: Table instance_objects (objects)

SID	ID	Name	Start_frame	End_frame	Children	Representative	Creator	App_info
44	1418	Flowers in parks	0	0	2	0	1	{"Other"}
47	1436	Rockery	0	0	0	0	1	{"Ivy"}
45	1426	Queens square	0	0	2	0	1	{"4.a",""}
47	1440	Title: Most actors are new	53902	54004	0	53943	1	{""}
46	1441	Credits	54478	54926	0	54902	1	{"E.a.A"}
47	1448	Susan starts her recollection	39347	39865	0	39838	1	{"Susan"}
47	1444	Leland starts recollection	22808	23885	0	22922	1	{"Leland"}
47	1452	MothercallsCharlesin, Thatcher tries to take him	9598	10477	0	9598	1	{"Mother, Boy Charles, Father, Thatcher"}
47	1457	Sometimes I would prefer a rival of flesh and blood	24386	24403	0	24394	1	{"Emily"}
45	138	Kane takes over Inquirer	15181	18265	2	15201	1	{"5.b","1895"}
45	1395	End	53902	54974	10	54904	1	{"E.a",""}
47	1465	Flustered Carter	16961	16984	0	16967	1	{"Carter"}
47	1469	Thompson meets Bernstein	13925	15181	0	13944	1	{"Thompson, Bernstein"}
47	1461	Kane decides to stay with Susan	31641	31994	0	31774	1	{"Kane, Gettys, Emily, Susan"}
47	1473	Raymond concludes	51262	51579	0	51517	1	{"Raymond, Thompson"}
44	236	Fifth flashback	49719	51262	1	51180	1	{"Flashback"}
46	70	Isolation at Xanadu	5476	5654	0	5476	1	{"2.a.L"}

Table 31: Table instance_objects (objects)

SID	ID	Source	Destination	Creator	GroupID	Name
5	87	55	59	1	55	Parent of
6	75	59	60	1	55	Next in sequence
5	88	55	60	1	55	Parent of
6	76	60	61	1	55	Next in sequence
5	89	55	61	1	55	Parent of
6	77	61	62	1	55	Next in sequence
5	90	55	62	1	55	Parent of
6	78	62	63	1	55	Next in sequence
5	91	55	63	1	55	Parent of
6	79	63	64	1	55	Next in sequence
5	92	55	64	1	55	Parent of
6	80	64	65	1	55	Next in sequence
5	93	55	65	1	55	Parent of
6	81	65	66	1	55	Next in sequence
5	94	55	66	1	55	Parent of
6	82	66	67	1	55	Next in sequence
5	95	55	67	1	55	Parent of
6	83	67	68	1	55	Next in sequence
5	96	55	68	1	55	Parent of
6	84	68	69	1	55	Next in sequence
5	97	55	69	1	55	Parent of
6	85	69	70	1	55	Next in sequence
5	98	55	70	1	55	Parent of
6	86	70	71	1	55	Next in sequence
5	99	55	71	1	55	Parent of
5	127	123	107	1	123	Parent of
6	124	107	108	1	123	Next in sequence
5	128	123	108	1	123	Parent of
6	125	108	109	1	123	Next in sequence
5	129	123	109	1	123	Parent of
6	126	109	110	1	123	Next in sequence
5	130	123	110	1	123	Parent of
5	145	136	138	1	136	Parent of
6	252	138	139	1	136	Next in sequence
5	146	136	139	1	136	Parent of
6	253	139	140	1	136	Next in sequence
5	147	136	140	1	136	Parent of
6	254	140	141	1	136	Next in sequence
5	148	136	141	1	136	Parent of
5	149	136	142	1	136	Parent of
5	150	134	135	1	134	Parent of
6	143	135	136	1	134	Next in sequence
5	151	134	136	1	134	Parent of
6	144	136	137	1	134	Next in sequence
5	152	134	137	1	134	Parent of
5	175	156	160	1	156	Parent of
5	179	158	161	1	158	Parent of
6	168	161	162	1	158	Next in sequence
5	180	158	162	1	158	Parent of
6	169	162	163	1	158	Next in sequence
5	181	158	163	1	158	Parent of
6	170	163	164	1	158	Next in sequence
5	182	158	164	1	158	Parent of
6	171	164	165	1	158	Next in sequence
5	183	158	165	1	158	Parent of
6	172	165	166	1	158	Next in sequence
5	184	158	166	1	158	Parent of
6	173	166	167	1	158	Next in sequence
5	185	158	167	1	158	Parent of
5	186	154	155	1	154	Parent of
6	174	155	156	1	154	Next in sequence
5	187	154	156	1	154	Parent of
6	176	156	157	1	154	Next in sequence
5	188	154	157	1	154	Parent of
6	177	157	158	1	154	Next in sequence
5	189	154	158	1	154	Parent of
6	178	158	159	1	154	Next in sequence
5	190	154	159	1	154	Parent of
5	215	194	195	1	194	Parent of
6	206	195	196	1	194	Next in sequence
5	216	194	196	1	194	Parent of
6	207	196	197	1	194	Next in sequence
5	217	194	197	1	194	Parent of
6	208	197	198	1	194	Next in sequence
5	218	194	198	1	194	Parent of
6	209	198	199	1	194	Next in sequence
5	219	194	199	1	194	Parent of

Table 32: Table instance_links (relationships)

SID	ID	Source	Destination	Creator	GroupID	Name
6	210	199	200	1	194	Next in sequence
5	220	194	200	1	194	Parent of
6	211	200	201	1	194	Next in sequence
5	221	194	201	1	194	Parent of
6	212	201	202	1	194	Next in sequence
5	222	194	202	1	194	Parent of
6	213	202	203	1	194	Next in sequence
5	223	194	203	1	194	Parent of
6	214	203	204	1	194	Next in sequence
5	224	194	204	1	194	Parent of
5	227	192	193	1	192	Parent of
6	226	193	194	1	192	Next in sequence
5	228	192	194	1	192	Parent of
6	225	194	205	1	192	Next in sequence
5	229	192	205	1	192	Parent of
5	239	236	238	1	236	Parent of
5	244	234	235	1	234	Parent of
6	237	235	236	1	234	Next in sequence
5	245	234	236	1	234	Parent of
6	241	236	240	1	234	Next in sequence
5	246	234	240	1	234	Parent of
6	243	240	242	1	234	Next in sequence
5	247	234	242	1	234	Parent of
5	52	48	49	1	48	Parent of
6	72	49	50	1	48	Next in sequence
5	53	48	50	1	48	Parent of
6	73	50	51	1	48	Next in sequence
5	54	48	51	1	48	Parent of
6	101	51	100	1	48	Next in sequence
5	102	48	100	1	48	Parent of
6	105	100	103	1	48	Next in sequence
5	104	48	103	1	48	Parent of
6	230	103	134	1	48	Next in sequence
5	153	48	134	1	48	Parent of
6	231	134	154	1	48	Next in sequence
5	191	48	154	1	48	Parent of
6	232	154	192	1	48	Next in sequence
5	233	48	192	1	48	Parent of
6	248	192	234	1	48	Next in sequence
5	249	48	234	1	48	Parent of
6	414	234	250	1	48	Next in sequence
5	251	48	250	1	48	Parent of
5	57	51	55	1	51	Parent of
6	74	55	56	1	51	Next in sequence
5	58	51	56	1	51	Parent of
5	121	103	111	1	103	Parent of
5	122	103	106	1	103	Parent of
6	131	106	123	1	103	Next in sequence
5	133	103	123	1	103	Parent of
6	415	123	111	1	103	Next in sequence
5	421	416	417	1	416	Parent of
6	422	417	418	1	416	Next in sequence
5	423	416	418	1	416	Parent of
6	424	418	419	1	416	Next in sequence
5	425	416	419	1	416	Parent of
5	427	416	420	1	416	Parent of
5	431	417	428	1	417	Parent of
6	432	428	429	1	417	Next in sequence
5	433	417	429	1	417	Parent of
6	434	429	430	1	417	Next in sequence
5	435	417	430	1	417	Parent of
5	439	418	436	1	418	Parent of
6	440	436	437	1	418	Next in sequence
5	441	418	437	1	418	Parent of
6	442	437	438	1	418	Next in sequence
5	443	418	438	1	418	Parent of
5	445	50	444	1	50	Parent of
5	446	428	444	1	428	Parent of
5	448	429	447	1	429	Parent of
5	450	142	449	1	142	Parent of
5	452	107	451	1	107	Parent of
5	454	238	453	1	238	Parent of
5	456	429	449	1	429	Parent of
5	458	429	451	1	429	Parent of
5	461	429	453	1	429	Parent of
6	457	449	451	1	429	Next in sequence
6	462	447	449	1	0	Next in sequence
6	459	451	453	1	429	Next in sequence

Table 32: Table instance_links (relationships)

SID	ID	Source	Destination	Creator	GroupID	Name
6	480	419	478	1	416	Next in sequence
5	482	416	478	1	416	Parent of
6	483	478	479	1	416	Next in sequence
6	484	479	420	1	416	Next in sequence
5	485	416	479	1	416	Parent of
5	486	419	464	1	419	Parent of
6	487	464	465	1	419	Next in sequence
5	488	419	465	1	419	Parent of
6	489	463	464	1	419	Next in sequence
5	490	419	463	1	419	Parent of
6	491	465	466	1	419	Next in sequence
5	492	419	466	1	419	Parent of
6	493	466	467	1	419	Next in sequence
5	494	419	467	1	419	Parent of
6	495	467	475	1	419	Next in sequence
5	496	419	475	1	419	Parent of
6	497	475	476	1	419	Next in sequence
5	498	419	476	1	419	Parent of
6	499	476	477	1	419	Next in sequence
5	500	419	477	1	419	Parent of
5	501	467	468	1	467	Parent of
6	502	468	469	1	467	Next in sequence
5	503	467	469	1	467	Parent of
6	504	469	470	1	467	Next in sequence
5	505	467	470	1	467	Parent of
6	506	470	471	1	467	Next in sequence
5	507	467	471	1	467	Parent of
6	508	471	472	1	467	Next in sequence
5	509	467	472	1	467	Parent of
6	510	472	473	1	467	Next in sequence
5	511	467	473	1	467	Parent of
6	512	473	474	1	467	Next in sequence
5	513	467	474	1	467	Parent of
5	517	436	514	1	436	Parent of
6	518	514	515	1	436	Next in sequence
5	519	436	515	1	436	Parent of
5	521	436	516	1	436	Parent of
5	525	436	522	1	436	Parent of
5	564	559	560	1	559	Parent of
5	566	559	561	1	559	Parent of
6	520	515	522	1	436	Next in sequence
5	527	140	526	1	140	Parent of
5	529	436	526	1	436	Parent of
6	528	526	516	1	436	Next in sequence
6	524	522	526	1	436	Next in sequence
5	531	437	530	1	437	Parent of
6	533	530	532	1	437	Next in sequence
5	534	437	532	1	437	Parent of
5	568	559	562	1	559	Parent of
5	537	437	535	1	437	Parent of
5	540	140	538	1	140	Parent of
5	542	437	538	1	437	Parent of
5	571	559	563	1	559	Parent of
5	544	138	543	1	138	Parent of
5	573	558	559	1	558	Parent of
5	547	437	543	1	437	Parent of
6	546	543	530	1	437	Next in sequence
6	545	532	535	1	437	Next in sequence
557	565	560	561	1	559	Next quarter turn
6	541	535	538	1	437	Next in sequence
557	567	561	562	1	559	Next quarter turn
557	570	562	563	1	559	Next quarter turn
557	572	563	560	1	0	Next quarter turn
5	576	574	575	1	574	Parent of
554	577	575	560	1	0	Entry
553	578	574	559	1	0	Meets
5	579	558	574	1	558	Parent of
5	582	558	580	1	558	Parent of
5	583	580	581	1	580	Parent of
553	584	580	559	1	0	Meets
555	585	560	581	1	0	Exit
553	587	580	586	1	0	Meets
5	588	558	586	1	558	Parent of
5	590	586	589	1	586	Parent of
557	592	589	591	1	586	Next quarter turn
5	593	586	591	1	586	Parent of
557	595	591	594	1	586	Next quarter turn
5	596	586	594	1	586	Parent of

Table 32: Table instance_links (relationships)

SID	ID	Source	Destination	Creator	GroupID	Name
557	598	597	589	1	586	Next quarter turn
5	599	586	597	1	586	Parent of
557	600	594	597	1	0	Next quarter turn
554	601	581	589	1	0	Entry
553	603	574	602	1	0	Meets
5	604	558	602	1	558	Parent of
5	606	602	605	1	602	Parent of
557	608	605	607	1	602	Next quarter turn
5	609	602	607	1	602	Parent of
557	611	607	610	1	602	Next quarter turn
5	612	602	610	1	602	Parent of
557	614	610	613	1	602	Next quarter turn
5	615	602	613	1	602	Parent of
557	616	613	605	1	0	Next quarter turn
555	617	613	575	1	0	Exit
5	619	558	618	1	558	Parent of
5	621	618	620	1	618	Parent of
555	622	589	620	1	0	Exit
553	623	618	586	1	0	Meets
5	626	558	624	1	558	Parent of
5	631	624	630	1	624	Parent of
557	632	630	627	1	624	Next quarter turn
5	633	624	627	1	624	Parent of
557	634	627	628	1	624	Next quarter turn
5	635	624	628	1	624	Parent of
557	636	629	630	1	624	Next quarter turn
5	637	624	629	1	624	Parent of
557	638	628	629	1	0	Next quarter turn
554	639	620	627	1	0	Entry
555	641	628	640	1	0	Exit
5	642	618	640	1	618	Parent of
554	643	640	597	1	0	Entry
5	645	580	644	1	580	Parent of
555	646	591	644	1	0	Exit
554	647	644	563	1	0	Entry
5	649	574	648	1	574	Parent of
555	650	561	648	1	0	Exit
554	651	648	610	1	0	Entry
553	655	654	602	1	0	Meets
553	658	618	624	1	0	Meets
553	659	654	624	1	0	Meets
5	660	656	653	1	656	Parent of
5	661	558	656	1	558	Parent of
555	662	630	653	1	0	Exit
553	663	656	624	1	0	Meets
5	668	657	666	1	657	Parent of
557	669	666	667	1	657	Next quarter turn
5	670	657	667	1	657	Parent of
557	671	667	664	1	657	Next quarter turn
5	672	657	664	1	657	Parent of
557	673	664	665	1	657	Next quarter turn
5	674	657	665	1	657	Parent of
557	675	665	666	1	0	Next quarter turn
553	676	657	656	1	0	Meets
5	677	558	657	1	558	Parent of
554	678	653	664	1	0	Entry
5	679	558	654	1	558	Parent of
5	687	682	685	1	682	Parent of
557	688	685	686	1	682	Next quarter turn
5	689	682	686	1	682	Parent of
557	690	686	683	1	682	Next quarter turn
5	691	682	683	1	682	Parent of
557	692	684	685	1	682	Next quarter turn
5	693	682	684	1	682	Parent of
557	694	683	684	1	0	Next quarter turn
553	695	680	682	1	0	Meets
553	696	680	657	1	0	Meets
5	697	680	681	1	680	Parent of
554	698	681	683	1	0	Entry
555	699	664	681	1	0	Exit
5	700	558	680	1	558	Parent of
5	701	558	682	1	558	Parent of
555	706	607	704	1	0	Exit
555	707	683	703	1	0	Exit
554	708	703	605	1	0	Entry
5	709	702	704	1	702	Parent of
5	710	702	703	1	702	Parent of
5	711	558	702	1	558	Parent of

Table 32: Table instance_links (relationships)

SID	ID	Source	Destination	Creator	GroupID	Name
553	712	702	682	1	0	Meets
553	713	702	602	1	0	Meets
5	715	680	714	1	680	Parent of
555	716	684	714	1	0	Exit
554	717	714	667	1	0	Entry
5	719	656	718	1	656	Parent of
555	720	665	718	1	0	Exit
554	721	718	629	1	0	Entry
554	724	723	630	1	0	Entry
555	725	605	723	1	0	Exit
554	726	722	613	1	0	Entry
555	727	627	722	1	0	Exit
5	728	654	722	1	654	Parent of
5	729	654	723	1	654	Parent of
554	730	704	686	1	0	Entry
5	739	733	737	1	733	Parent of
557	740	737	734	1	733	Next quarter turn
5	741	733	734	1	733	Parent of
557	742	734	735	1	733	Next quarter turn
5	743	733	735	1	733	Parent of
557	744	735	736	1	733	Next quarter turn
5	745	733	736	1	733	Parent of
557	746	736	737	1	0	Next quarter turn
555	747	629	732	1	0	Exit
554	748	732	734	1	0	Entry
554	749	738	628	1	0	Entry
555	750	735	738	1	0	Exit
5	751	558	731	1	558	Parent of
553	752	731	624	1	0	Meets
5	753	731	738	1	731	Parent of
5	754	731	732	1	731	Parent of
553	755	731	733	1	0	Meets
5	756	558	733	1	558	Parent of
5	760	758	628	1	758	Parent of
5	762	758	640	1	758	Parent of
5	764	758	703	1	758	Parent of
5	765	758	605	1	758	Parent of
5	767	758	723	1	758	Parent of
5	770	758	630	1	758	Parent of
5	773	758	627	1	758	Parent of
6	763	628	640	1	758	Next in sequence
6	766	703	605	1	758	Next in sequence
6	768	605	723	1	758	Next in sequence
6	769	723	630	1	758	Next in sequence
6	771	630	627	1	758	Next in sequence
6	772	627	628	1	758	Next in sequence
5	826	822	823	1	822	Parent of
6	827	823	824	1	822	Next in sequence
5	828	822	824	1	822	Parent of
5	830	822	825	1	822	Parent of
5	836	832	833	1	832	Parent of
6	837	833	834	1	832	Next in sequence
5	838	832	834	1	832	Parent of
5	840	832	835	1	832	Parent of
5	846	842	843	1	842	Parent of
6	847	843	844	1	842	Next in sequence
5	848	842	844	1	842	Parent of
6	849	844	845	1	842	Next in sequence
5	850	842	845	1	842	Parent of
5	856	852	853	1	852	Parent of
6	857	853	854	1	852	Next in sequence
5	858	852	854	1	852	Parent of
5	860	852	855	1	852	Parent of
5	866	862	863	1	862	Parent of
6	867	863	864	1	862	Next in sequence
5	868	862	864	1	862	Parent of
6	869	864	865	1	862	Next in sequence
5	870	862	865	1	862	Parent of
5	876	872	873	1	872	Parent of
6	877	873	874	1	872	Next in sequence
5	878	872	874	1	872	Parent of
6	879	874	875	1	872	Next in sequence
5	880	872	875	1	872	Parent of
5	886	882	883	1	882	Parent of
6	887	883	884	1	882	Next in sequence
5	888	882	884	1	882	Parent of
5	890	882	885	1	882	Parent of
5	920	918	804	1	918	Parent of

Table 32: Table instance_links (relationships)

SID	ID	Source	Destination	Creator	GroupID	Name
6	921	804	805	1	918	Next in sequence
5	922	918	805	1	918	Parent of
6	923	805	919	1	918	Next in sequence
5	924	918	919	1	918	Parent of
6	925	919	806	1	918	Next in sequence
5	926	918	806	1	918	Parent of
5	949	897	900	1	897	Parent of
6	950	900	902	1	897	Next in sequence
5	951	897	902	1	897	Parent of
6	952	902	904	1	897	Next in sequence
5	953	897	904	1	897	Parent of
6	954	904	906	1	897	Next in sequence
5	955	897	906	1	897	Parent of
5	956	936	931	1	936	Parent of
6	957	931	933	1	936	Next in sequence
5	958	936	933	1	936	Parent of
6	959	929	931	1	936	Next in sequence
5	960	936	929	1	936	Parent of
6	961	933	935	1	936	Next in sequence
5	962	936	935	1	936	Parent of
5	981	783	918	1	783	Parent of
5	982	783	897	1	783	Parent of
5	983	783	936	1	783	Parent of
5	984	783	822	1	783	Parent of
5	985	783	832	1	783	Parent of
5	986	783	842	1	783	Parent of
5	987	783	852	1	783	Parent of
5	988	783	862	1	783	Parent of
5	989	783	872	1	783	Parent of
5	990	783	882	1	783	Parent of
5	1019	1017	1002	1	1017	Parent of
5	1020	1017	1004	1	1017	Parent of
5	1021	1017	1006	1	1017	Parent of
5	1022	1018	1017	1	1018	Parent of
5	1023	1016	1001	1	1016	Parent of
5	1024	1016	1003	1	1016	Parent of
5	1025	1016	1005	1	1016	Parent of
5	1026	1016	1007	1	1016	Parent of
5	1027	1016	1008	1	1016	Parent of
5	1028	1016	1009	1	1016	Parent of
5	1029	1016	1010	1	1016	Parent of
5	1030	1016	1011	1	1016	Parent of
5	1031	1016	1012	1	1016	Parent of
5	1032	1016	1013	1	1016	Parent of
5	1033	1016	1014	1	1016	Parent of
5	1034	1016	1015	1	1016	Parent of
5	1035	992	1016	1	992	Parent of
5	1036	992	1018	1	992	Parent of
991	1037	897	1018	1	0	Uses
991	1038	897	1016	1	0	Uses
5	1048	1047	1039	1	1047	Parent of
5	1049	1047	1040	1	1047	Parent of
5	1050	1047	1041	1	1047	Parent of
5	1051	1046	1042	1	1046	Parent of
5	1052	1045	1047	1	1045	Parent of
5	1053	1045	1046	1	1045	Parent of
5	1054	1044	1043	1	1044	Parent of
5	1055	992	1045	1	992	Parent of
5	1056	992	1044	1	992	Parent of
991	1057	918	1045	1	0	Uses
991	1058	918	1044	1	0	Uses
5	1073	1070	1066	1	1070	Parent of
5	1074	1071	1064	1	1071	Parent of
5	1075	1071	1062	1	1071	Parent of
5	1076	1072	1061	1	1072	Parent of
5	1077	1072	1059	1	1072	Parent of
5	1078	1068	1060	1	1068	Parent of
5	1079	1068	1063	1	1068	Parent of
5	1080	1068	1065	1	1068	Parent of
5	1081	1068	1067	1	1068	Parent of
5	1082	1069	1072	1	1069	Parent of
5	1083	1069	1071	1	1069	Parent of
5	1084	1069	1070	1	1069	Parent of
5	1085	992	1069	1	992	Parent of
5	1086	992	1068	1	992	Parent of
991	1087	936	1069	1	0	Uses
991	1088	936	1068	1	0	Uses
6	829	1089	825	1	822	Next in sequence

Table 32: Table instance_links (relationships)

SID	ID	Source	Destination	Creator	GroupID	Name
5	1090	822	1089	1	822	Parent of
6	1091	824	1089	1	0	Next in sequence
5	1101	1100	1092	1	1100	Parent of
5	1102	1100	1094	1	1100	Parent of
5	1103	1100	1096	1	1100	Parent of
5	1104	1099	1100	1	1099	Parent of
5	1105	1098	1093	1	1098	Parent of
5	1106	1098	1095	1	1098	Parent of
5	1107	1098	1097	1	1098	Parent of
5	1108	992	1098	1	992	Parent of
5	1109	992	1099	1	992	Parent of
991	1110	822	1099	1	0	Uses
991	1111	822	1098	1	0	Uses
6	839	1113	835	1	832	Next in sequence
6	1114	834	1112	1	0	Next in sequence
6	1115	1112	1113	1	0	Next in sequence
5	1116	832	1112	1	832	Parent of
5	1117	832	1113	1	832	Parent of
5	1126	1124	1118	1	1124	Parent of
5	1127	1124	1119	1	1124	Parent of
5	1128	1125	1120	1	1125	Parent of
5	1129	1124	1121	1	1124	Parent of
5	1130	1124	1122	1	1124	Parent of
5	1131	1125	1123	1	1125	Parent of
5	1133	1132	1124	1	1132	Parent of
5	1134	1132	1125	1	1132	Parent of
5	1135	992	1132	1	992	Parent of
991	1136	832	1132	1	0	Uses
5	1140	1138	1137	1	1138	Parent of
5	1141	1139	1138	1	1139	Parent of
5	1142	992	1139	1	992	Parent of
991	1143	842	1139	1	0	Uses
6	859	1144	855	1	852	Next in sequence
6	1145	854	1144	1	0	Next in sequence
5	1146	852	1144	1	852	Parent of
5	1154	1147	1153	1	1147	Parent of
5	1155	1147	1148	1	1147	Parent of
5	1158	1156	1150	1	1156	Parent of
5	1159	1156	1151	1	1156	Parent of
5	1160	1157	1152	1	1157	Parent of
5	1161	1149	1156	1	1149	Parent of
5	1162	1149	1157	1	1149	Parent of
991	1163	852	1147	1	0	Uses
991	1164	852	1149	1	0	Uses
5	1165	992	1149	1	992	Parent of
5	1166	992	1147	1	992	Parent of
5	1171	1170	1167	1	1170	Parent of
5	1172	1170	1168	1	1170	Parent of
5	1173	1170	1169	1	1170	Parent of
5	1174	992	1170	1	992	Parent of
991	1175	862	1170	1	0	Uses
5	1185	1182	1176	1	1182	Parent of
5	1186	1180	1177	1	1180	Parent of
5	1187	1183	1178	1	1183	Parent of
5	1188	1181	1182	1	1181	Parent of
5	1189	1181	1180	1	1181	Parent of
5	1190	1181	1183	1	1181	Parent of
5	1191	1184	1179	1	1184	Parent of
5	1192	992	1184	1	992	Parent of
5	1193	992	1181	1	992	Parent of
991	1194	872	1181	1	0	Uses
991	1195	872	1184	1	0	Uses
5	1210	1208	1197	1	1208	Parent of
5	1211	1208	1198	1	1208	Parent of
5	1212	1208	1201	1	1208	Parent of
5	1213	1209	1199	1	1209	Parent of
5	1214	1209	1202	1	1209	Parent of
5	1215	1207	1204	1	1207	Parent of
5	1216	1206	1207	1	1206	Parent of
5	1217	1206	1209	1	1206	Parent of
5	1218	1206	1208	1	1206	Parent of
5	1219	1205	1196	1	1205	Parent of
5	1220	1205	1200	1	1205	Parent of
5	1221	1205	1203	1	1205	Parent of
5	1222	992	1205	1	992	Parent of
5	1223	992	1206	1	992	Parent of
991	1224	882	1206	1	0	Uses
991	1225	882	1205	1	0	Uses

Table 32: Table instance_links (relationships)

SID	ID	Source	Destination	Creator	GroupID	Name
6	889	1226	885	1	882	Next in sequence
6	1227	884	1226	1	0	Next in sequence
5	1228	882	1226	1	882	Parent of
5	1233	1229	1230	1	1229	Parent of
5	1234	1229	1231	1	1229	Parent of
5	1235	1229	1232	1	1229	Parent of
6	1236	1230	1231	1	0	Next in sequence
6	1237	1231	1232	1	0	Next in sequence
5	1242	1240	1241	1	1240	Parent of
5	1243	1238	1240	1	1238	Parent of
5	1245	1239	1244	1	1239	Parent of
5	1246	992	1239	1	992	Parent of
5	1247	992	1238	1	992	Parent of
991	1248	1229	1238	1	0	Uses
991	1249	1229	1239	1	0	Uses
5	1250	783	1229	1	783	Parent of
5	1255	1251	1252	1	1251	Parent of
5	1256	1251	1253	1	1251	Parent of
5	1257	1251	1254	1	1251	Parent of
6	1258	1252	1253	1	0	Next in sequence
6	1259	1253	1254	1	0	Next in sequence
5	1265	1264	1263	1	1264	Parent of
5	1266	1261	1262	1	1261	Parent of
5	1267	1260	1264	1	1260	Parent of
5	1268	1260	1261	1	1260	Parent of
991	1269	1251	1260	1	0	Uses
5	1270	783	1251	1	783	Parent of
5	1271	992	1260	1	992	Parent of
5	1276	1272	1273	1	1272	Parent of
5	1277	1272	1274	1	1272	Parent of
5	1278	1272	1275	1	1272	Parent of
6	1279	1273	1274	1	0	Next in sequence
6	1280	1274	1275	1	0	Next in sequence
5	1288	1283	1284	1	1283	Parent of
5	1289	1287	1286	1	1287	Parent of
5	1290	1281	1283	1	1281	Parent of
5	1291	1281	1287	1	1281	Parent of
5	1292	1282	1285	1	1282	Parent of
5	1293	992	1282	1	992	Parent of
5	1294	992	1281	1	992	Parent of
5	1295	783	1272	1	783	Parent of
991	1296	1272	1281	1	0	Uses
991	1297	1272	1282	1	0	Uses
5	1302	1298	1299	1	1298	Parent of
6	1303	1299	1300	1	1298	Next in sequence
5	1304	1298	1300	1	1298	Parent of
6	1305	1300	1301	1	1298	Next in sequence
5	1306	1298	1301	1	1298	Parent of
5	1307	783	1298	1	783	Parent of
5	1315	1309	1314	1	1309	Parent of
5	1316	1309	1313	1	1309	Parent of
5	1317	1310	1311	1	1310	Parent of
5	1318	1310	1312	1	1310	Parent of
5	1319	1308	1309	1	1308	Parent of
5	1320	1308	1310	1	1308	Parent of
5	1321	992	1308	1	992	Parent of
991	1322	1298	1308	1	0	Uses
5	1327	1323	1324	1	1323	Parent of
6	1328	1324	1325	1	1323	Next in sequence
5	1329	1323	1325	1	1323	Parent of
6	1330	1325	1326	1	1323	Next in sequence
5	1331	1323	1326	1	1323	Parent of
5	1332	783	1323	1	783	Parent of
5	1342	1334	1341	1	1334	Parent of
5	1343	1334	1340	1	1334	Parent of
5	1344	1335	1337	1	1335	Parent of
5	1345	1336	1338	1	1336	Parent of
5	1346	1336	1339	1	1336	Parent of
5	1347	1333	1336	1	1333	Parent of
5	1348	1333	1335	1	1333	Parent of
5	1349	992	1333	1	992	Parent of
5	1350	992	1334	1	992	Parent of
991	1351	1323	1333	1	0	Uses
991	1352	1323	1334	1	0	Uses
5	1358	1353	1354	1	1353	Parent of
6	1359	1354	1355	1	1353	Next in sequence
5	1360	1353	1355	1	1353	Parent of
6	1361	1355	1356	1	1353	Next in sequence

Table 32: Table instance_links (relationships)

SID	ID	Source	Destination	Creator	GroupID	Name
5	1362	1353	1356	1	1353	Parent of
6	1363	1356	1357	1	1353	Next in sequence
5	1364	1353	1357	1	1353	Parent of
5	1365	783	1353	1	783	Parent of
5	1377	1368	1369	1	1368	Parent of
5	1378	1366	1368	1	1366	Parent of
5	1379	1370	1371	1	1370	Parent of
5	1380	1366	1370	1	1366	Parent of
5	1381	1372	1373	1	1372	Parent of
5	1382	1366	1372	1	1366	Parent of
5	1383	992	1366	1	992	Parent of
5	1384	1367	1374	1	1367	Parent of
5	1385	1367	1375	1	1367	Parent of
5	1386	1367	1376	1	1367	Parent of
5	1387	992	1367	1	992	Parent of
991	1388	1353	1366	1	0	Uses
991	1389	1353	1367	1	0	Uses
5	1391	1390	522	1	1390	Parent of
5	1392	49	1390	1	49	Parent of
5	1394	100	1393	1	100	Parent of
5	1396	250	1395	1	250	Parent of
5	1405	1397	1399	1	1397	Parent of
6	1406	1399	1400	1	1397	Next in sequence
5	1407	1397	1400	1	1397	Parent of
5	1409	1397	1401	1	1397	Parent of
6	1410	1400	1401	1	0	Next in sequence
6	1411	1401	1402	1	0	Next in sequence
6	1412	1402	1403	1	0	Next in sequence
6	1413	1403	1404	1	0	Next in sequence
5	1414	1397	1402	1	1397	Parent of
5	1415	1397	1403	1	1397	Parent of
5	1416	1397	1404	1	1397	Parent of
5	1421	1400	1420	1	1400	Parent of
5	1422	1401	1419	1	1401	Parent of
5	1423	1402	1417	1	1402	Parent of
5	1424	1402	1418	1	1402	Parent of
6	1425	1417	1418	1	0	Next in sequence
5	1430	1417	1426	1	1417	Parent of
5	1431	1417	1427	1	1417	Parent of
5	1432	1418	1428	1	1418	Parent of
5	1433	1418	1429	1	1418	Parent of
6	1434	1426	1427	1	0	Next in sequence
6	1435	1428	1429	1	0	Next in sequence
5	1437	1426	1436	1	1426	Parent of
5	1438	1426	591	1	1426	Parent of
6	1439	1436	591	1	0	Next in sequence
5	1442	1395	1440	1	1395	Parent of
5	1443	1395	1441	1	1395	Parent of
5	1445	155	1444	1	155	Parent of
5	1446	1395	1444	1	1395	Parent of
6	1447	1440	1444	1	0	Next in sequence
5	1449	193	1448	1	193	Parent of
5	1450	1395	1448	1	1395	Parent of
6	1451	1444	1448	1	0	Next in sequence
5	1453	107	1452	1	107	Parent of
6	1454	451	1452	1	0	Next in sequence
5	1455	1395	1452	1	1395	Parent of
6	1456	1448	1452	1	0	Next in sequence
5	1458	160	1457	1	160	Parent of
6	1459	1452	1457	1	0	Next in sequence
5	1460	1395	1457	1	1395	Parent of
5	1462	163	1461	1	163	Parent of
6	1463	1457	1461	1	0	Next in sequence
5	1464	1395	1461	1	1395	Parent of
5	1466	138	1465	1	138	Parent of
6	1467	1461	1465	1	0	Next in sequence
5	1468	1395	1465	1	1395	Parent of
5	1470	135	1469	1	135	Parent of
6	1471	1465	1469	1	0	Next in sequence
5	1472	1395	1469	1	1395	Parent of
5	1474	240	1473	1	240	Parent of
5	1475	1395	1473	1	1395	Parent of
6	1476	1473	1441	1	0	Next in sequence
6	1477	1469	1473	1	0	Next in sequence
6	255	141	142	1	136	Next in sequence

Table 32: Table instance_links (relationships)

SID	Name	Reverse_name	Source_def	Dest_def	Author	Cardinality
5	Parent of	Child of	548	549	1	m
5	Parent of	Child of	548	551	1	m
5	Parent of	Child of	42	43	1	m
5	Parent of	Child of	43	45	1	m
5	Parent of	Child of	43	44	1	m
5	Parent of	Child of	44	45	1	m
5	Parent of	Child of	45	46	1	m
5	Parent of	Child of	46	47	1	m
5	Parent of	Child of	45	47	1	m
5	Parent of	Child of	779	780	1	m
5	Parent of	Child of	779	781	1	m
5	Parent of	Child of	779	782	1	m
5	Parent of	Child of	966	967	1	m
5	Parent of	Child of	966	968	1	m
5	Parent of	Child of	966	969	1	m
5	Parent of	Child of	966	970	1	m
5	Parent of	Child of	971	972	1	m
5	Parent of	Child of	972	973	1	m
5	Parent of	Child of	971	974	1	m
5	Parent of	Child of	974	975	1	m
5	Parent of	Child of	974	976	1	m
5	Parent of	Child of	977	979	1	m
5	Parent of	Child of	977	978	1	m
5	Parent of	Child of	971	977	1	m
5	Parent of	Child of	980	964	1	m
5	Parent of	Child of	980	965	1	m
5	Parent of	Child of	980	966	1	m
5	Parent of	Child of	980	971	1	m
5	Parent of	Child of	774	779	1	m
5	Parent of	Child of			0	m
553	Meets	Meets	549	551	1	1
6	Next in sequence	Previous in sequence	780	781	1	1
6	Next in sequence	Previous in sequence	781	782	1	1
6	Next in sequence	Previous in sequence			0	1
553	Meets	Meets			1	1
5	Parent of	Child of	549	550	1	2
5	Parent of	Child of	551	552	1	4
554	Entry		550	552	1	1
554	Entry				1	1
555	Exit		552	550	1	1
555	Exit				1	1
557	Next quarter turn	Previous quarter turn			1	1
991	Uses	Uses			1	1
991	Uses	Uses	779	966	1	1
991	Uses	Uses	779	971	1	1

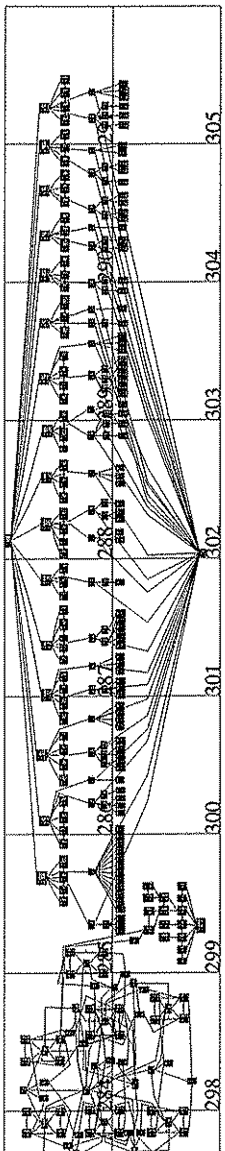
Table 33: Table schema_links (relationship types)

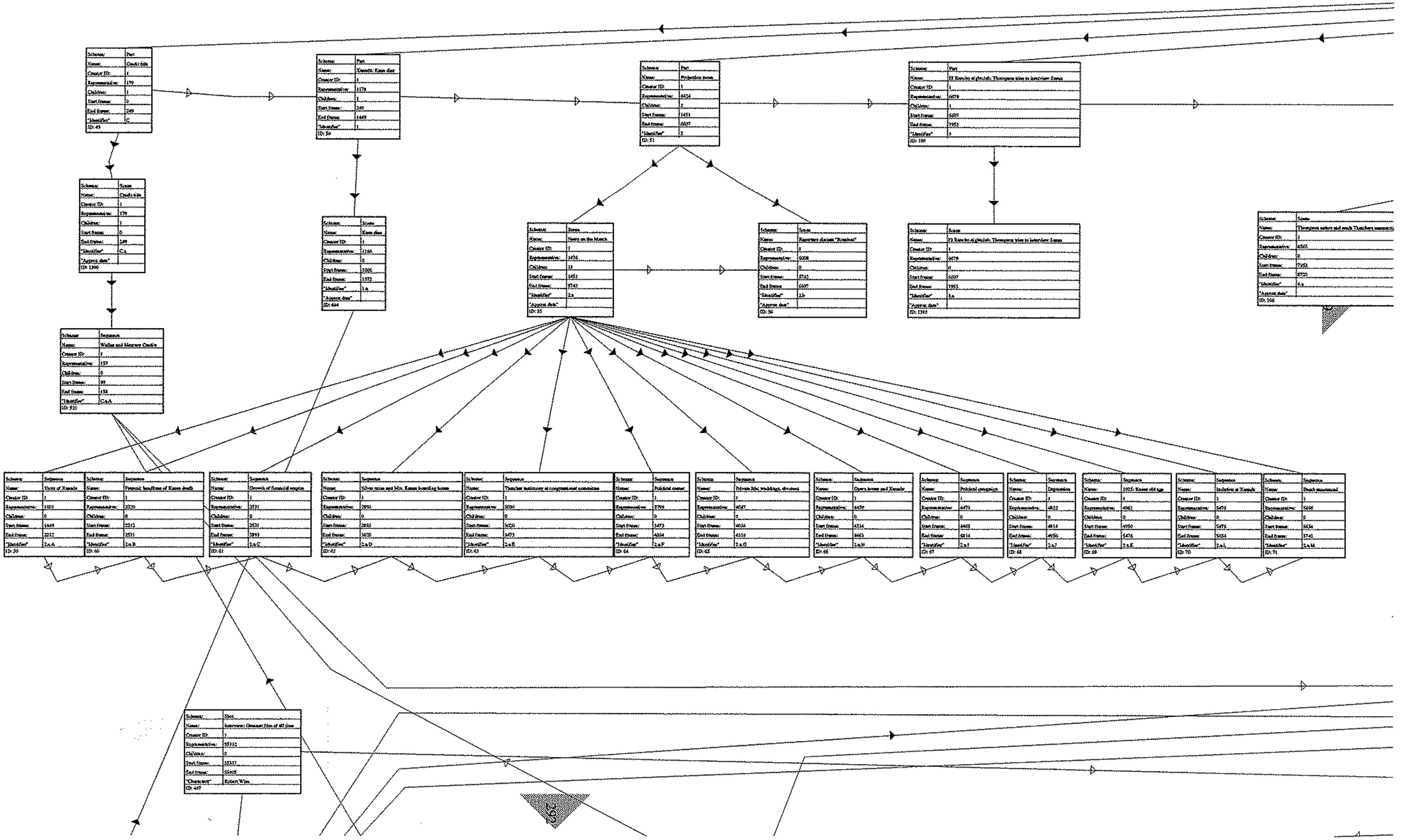
SID	Name	Kind	Author	Temporal	Indexed_children	Ordered_children	App_info	Ordering	App_fields
1	Indexed collection	Indexed collection	0	0	5	-1	{}	0	{}
0	Unit	Unit	0	1	-1	-1	{}	0	{}
4	Temporal collection	Temporal collection	0	1	5	6	{}	x	{}
3	Ordered collection	Ordered collection	0	0	5	6	{}	x	{}
47	Shot	Unit	1	1	-1	-1	{"Characters"}	0	{"<text>"}
551	Junction	Ordered collection	1	0	5	557	{}	n	{}
757	User Ordered Object	Ordered collection	1	0	5	6	{}	u	{}
548	Video-map	Indexed collection	1	0	5	-1	{}	0	{}
549	Journey	Indexed collection	1	0	5	-1	{}	0	{}
780	From	Unit	1	1	-1	-1	{}	0	{}
781	Via	Unit	1	1	-1	-1	{}	0	{}
782	To	Unit	1	1	-1	-1	{}	0	{}
964	Misenscene	Indexed collection	1	0	5	-1	{}	0	{}
965	Sound	Indexed collection	1	0	5	-1	{}	0	{}
774	Uses of Patterns of Movie style	Indexed collection	1	0	5	-1	{}	0	{}
966	Editing	Indexed collection	1	0	5	-1	{}	0	{}
968	Dissolve	Unit	1	1	-1	-1	{}	0	{}
969	Wipe	Unit	1	1	-1	-1	{}	0	{}
970	Cut	Unit	1	1	-1	-1	{}	0	{}
971	Cinematography	Indexed collection	1	0	5	-1	{}	0	{}
980	Techniques	Indexed collection	1	0	5	-1	{}	0	{}
42	Movie	Ordered collection	1	1	5	6	{"Bibliography"}	x	{"<text>"}
43	Part	Ordered collection	1	1	5	6	{"Identifier"}	x	{"<number>"}
44	Sub-Part	Ordered collection	1	1	5	6	{"Type"}	x	{"Flashback | Other"}
550	Flow	Unit	1	1	-1	-1	{"Heading"}	0	{"N|S|E|W"}
552	Clockwise Turn	Unit	1	1	-1	-1	{"Direction"}	0	{"NtoE|EtoS|StoW|WtoN"}
967	Fade	Unit	1	1	-1	-1	{"Type:"}	0	{"In|Out"}
973	Focal depth	Unit	1	1	-1	-1	{"Zoom:"}	0	{"In|Out"}
975	Pan	Unit	1	1	-1	-1	{"Direction:"}	0	{"Left|Right"}
976	Tilt	Unit	1	1	-1	-1	{"Direction:"}	0	{"Clockwise|Anti-clockwise"}
978	Height	Unit	1	1	-1	-1	{"Direction:"}	0	{"Up|Down"}
979	Motion	Unit	1	1	-1	-1	{"Direction:"}	0	{"In|Out"}
972	Lens change	Indexed collection	1	0	5	-1	{}	0	{}
974	Tripod change	Indexed collection	1	0	5	-1	{}	0	{}
977	Truck change	Indexed collection	1	0	5	-1	{}	0	{}
779	Spatial movement	Ordered collection	1	1	5	6	{"Type:","Effect:"}	x	{"Ingress|Recession","<text>"}
45	Scene	Ordered collection	1	1	5	6	{"Identifier","Approx date"}	x	{"<value[Child_o","Par"}
46	Sequence	Ordered collection	1	1	5	6	{"Identifier"}	x	{"<value[Child_o"}

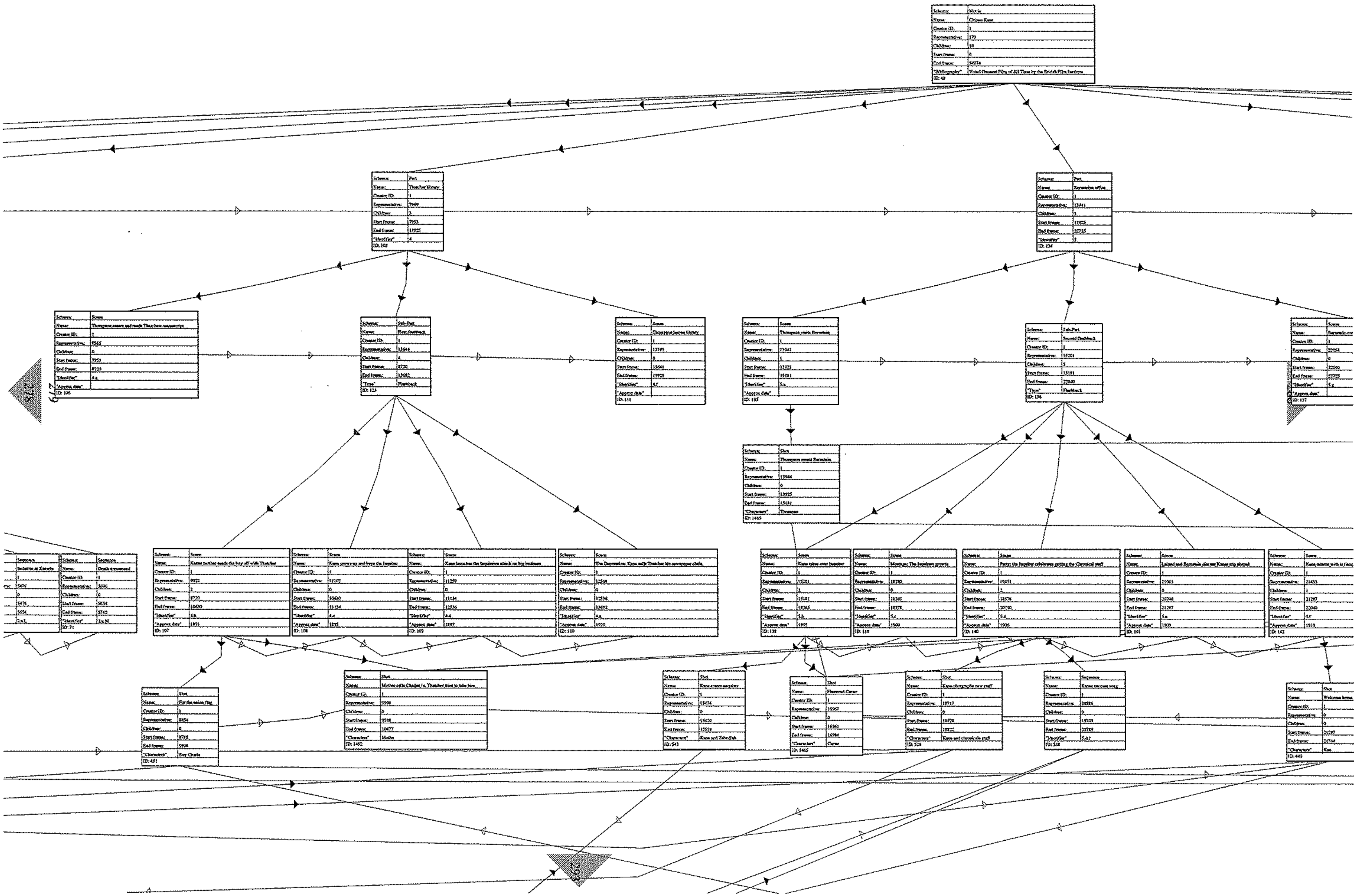
Table 34: Table schema_objects (object types)

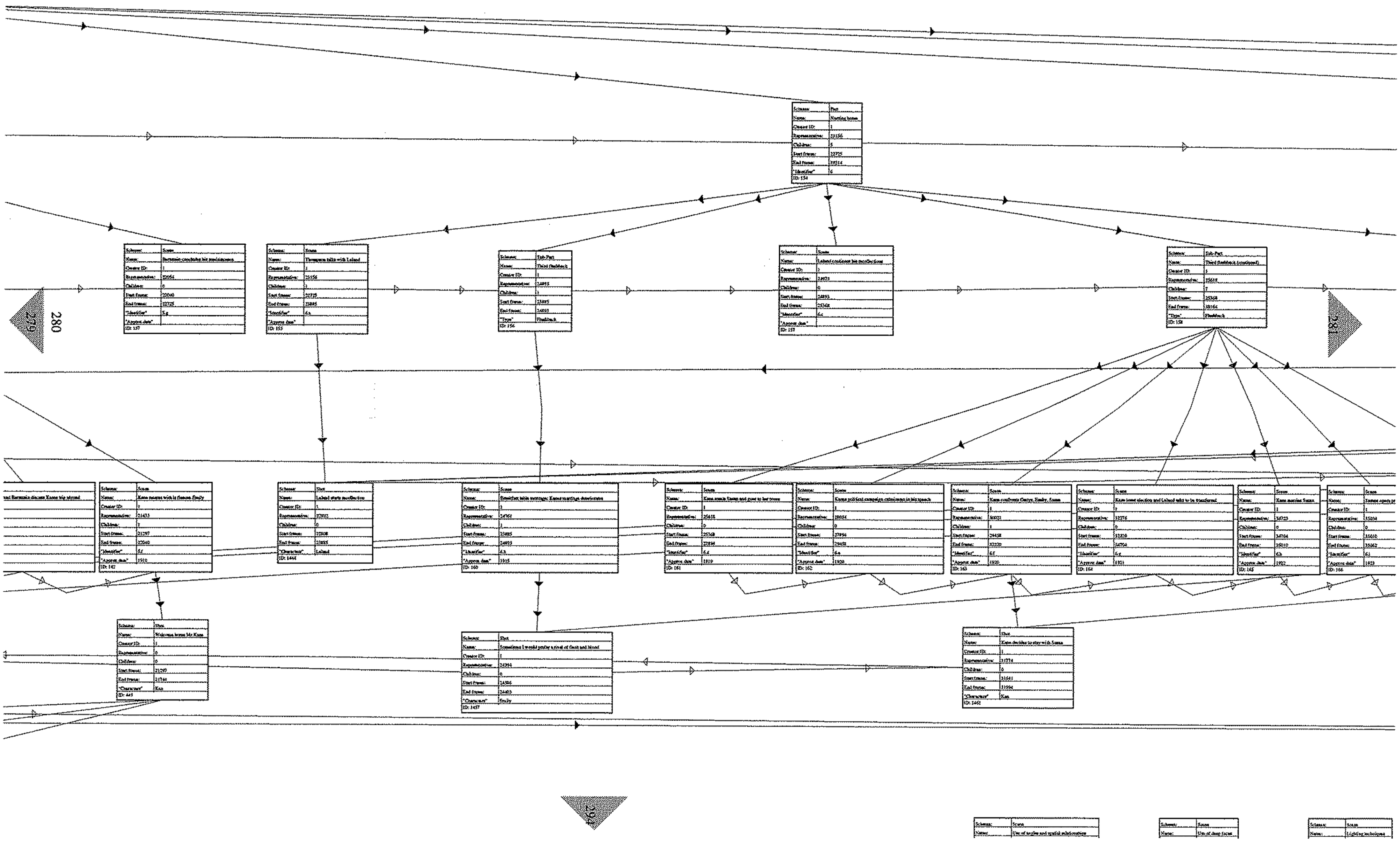
Appendix D VCMF Instances

This appendix gives a graphical representation of the instances used in the demonstration. The next page contains an overview of the rest of the appendix; the diagram is divided into squares which represent each page (indicated by a page number). On further pages grey triangles indicate appropriate adjacent pages.

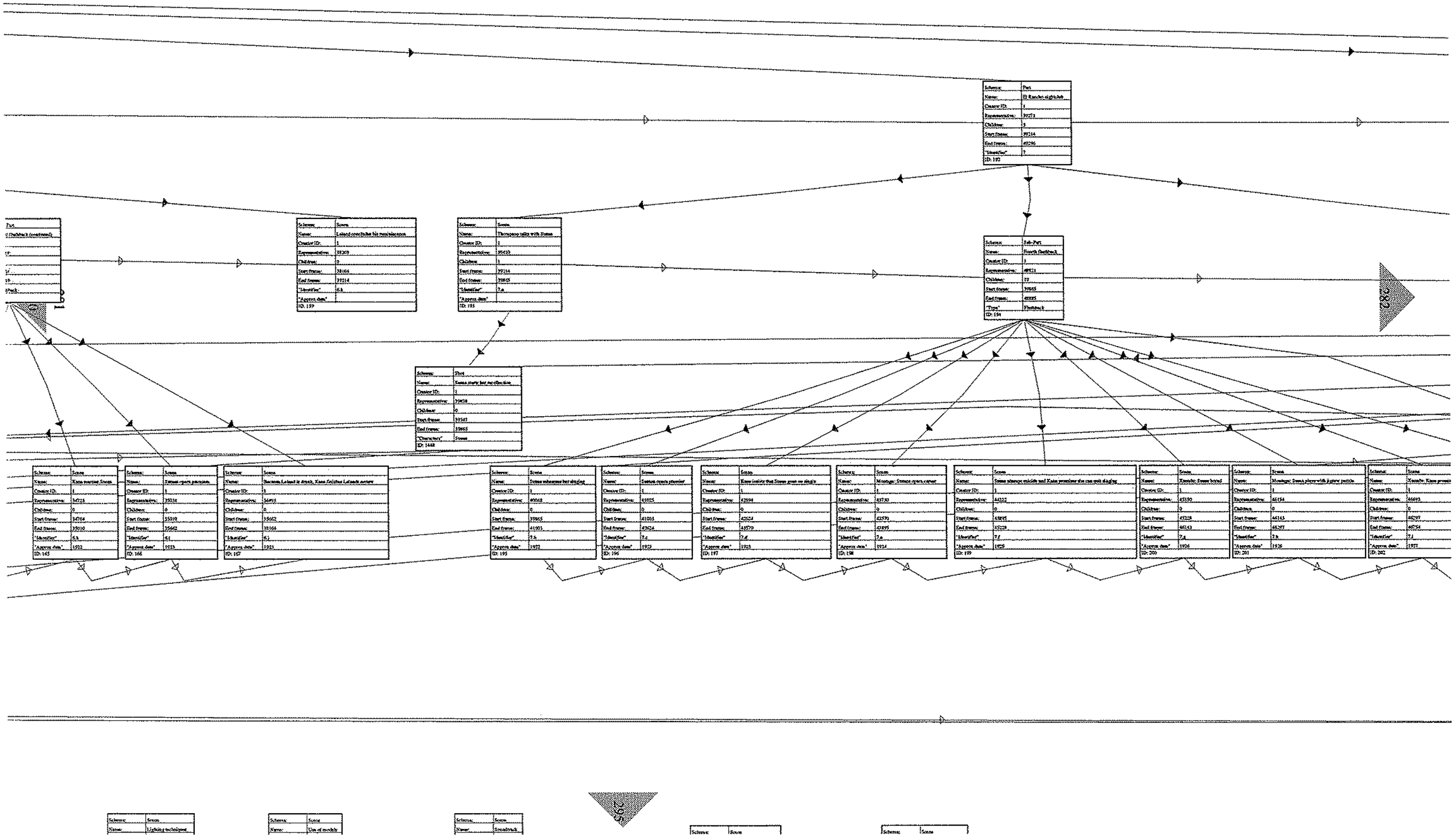

292
293
294
295
296
297
298
299
300
301
302
303
304
305
280
282
283
288

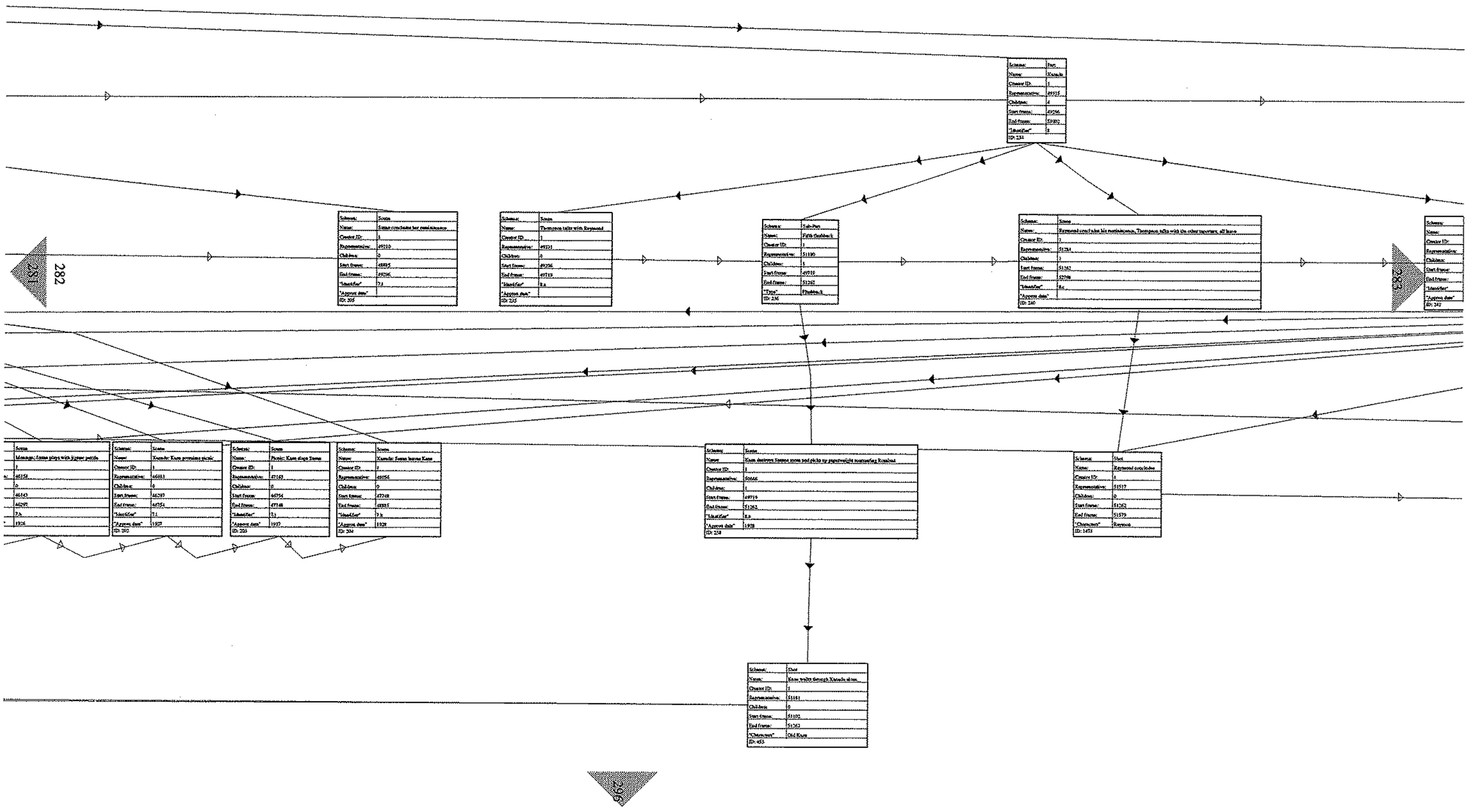

Schema: Part
Name: Credit title
Creator ID: 1
Representative: 179
Children: 1
Start frame: 0
End frame: 249
"Identifier" C
ID: 49
Schema: Scene
Name: Credit title
Creator ID: 1
Representative: 179
Children: 1
Start frame: 0
End frame: 249
"Identifier" C.a
"Approx date"
ID: 1390
Schema: Sequence
Name: Wallace and Mercury Credits
Creator ID: 1
Representative: 157
Children: 0
Start frame: 99
End frame: 158
"Identifier" C.a.A
ID: 522
Schema: Part
Name: Xanadu: Kane dies
Creator ID: 1
Representative: 1178
Children: 1
Start frame: 249
End frame: 1449
"Identifier" 1
ID: 50
Schema: Scene
Name: Kane dies
Creator ID: 1
Representative: 1166
Children: 0
Start frame: 1006
End frame: 1372
"Identifier" 1.a
"Approx date"
ID: 444
Schema: Part
Name: Projection room
Creator ID: 1
Representative: 6424
Children: 2
Start frame: 1451
End frame: 6607
"Identifier" 2
ID: 51
Schema: Part
Name: El Rancho at ghts/ds: Thompson tries to interview Susan
Creator ID: 1
Representative: 6679
Children: 1
Start frame: 6607
End frame: 7953
"Identifier" 3
ID: 100
Schema: Scene
Name: News on the March
Creator ID: 1
Representative: 1436
Children: 13
Start frame: 1451
End frame: 5742
"Identifier" 2.a
"Approx date"
ID: 55
Schema: Scene
Name: Reporters discuss "Rosebud"
Creator ID: 1
Representative: 6008
Children: 0
Start frame: 5742
End frame: 6607
"Identifier" 2.b
"Approx date"
ID: 56
Schema: Scene
Name: El Rancho at ghts/ds: Thompson tries to interview Susan
Creator ID: 1
Representative: 6679
Children: 0
Start frame: 6607
End frame: 7953
"Identifier" 3.a
"Approx date"
ID: 1393
Schema: Scene
Name: Thompson enters and reads Thatchers manuscri
Creator ID: 1
Representative: 8565
Children: 0
Start frame: 7953
End frame: 8720
"Identifier" 4.a
"Approx date"
ID: 106
Schema: Sequence
Name: Shots of Xanadu
Creator ID: 1
Representative: 1601
Children: 0
Start frame: 1449
End frame: 2212
"Identifier" 2.a.A
ID: 59
Schema: Sequence
Name: Funeral: headlines of Kanes death
Creator ID: 1
Representative: 2220
Children: 0
Start frame: 2212
End frame: 2531
"Identifier" 2.a.B
ID: 60
Schema: Sequence
Name: Growth of financial empire
Creator ID: 1
Representative: 2531
Children: 0
Start frame: 2531
End frame: 2893
"Identifier" 2.a.C
ID: 61
Schema: Sequence
Name: Silver mine and Mrs. Kanes boarding house
Creator ID: 1
Representative: 2893
Children: 0
Start frame: 2893
End frame: 3020
"Identifier" 2.a.D
ID: 62
Schema: Sequence
Name: Thatcher testimony at congressional committee
Creator ID: 1
Representative: 3030
Children: 0
Start frame: 3020
End frame: 3473
"Identifier" 2.a.E
ID: 63
Schema: Sequence
Name: Political career
Creator ID: 1
Representative: 3799
Children: 0
Start frame: 3473
End frame: 4034
"Identifier" 2.a.F
ID: 64
Schema: Sequence
Name: Private life; weddings, divorces
Creator ID: 1
Representative: 4047
Children: 0
Start frame: 4034
End frame: 4314
"Identifier" 2.a.G
ID: 65
Schema: Sequence
Name: Opera house and Xanadu
Creator ID: 1
Representative: 4459
Children: 0
Start frame: 4314
End frame: 4463
"Identifier" 2.a.H
ID: 66
Schema: Sequence
Name: Political campaigns
Creator ID: 1
Representative: 4473
Children: 0
Start frame: 4463
End frame: 4814
"Identifier" 2.a.I
ID: 67
Schema: Sequence
Name: Depression
Creator ID: 1
Representative: 4822
Children: 0
Start frame: 4814
End frame: 4956
"Identifier" 2.a.J
ID: 68
Schema: Sequence
Name: 1935: Kanes old age
Creator ID: 1
Representative: 4961
Children: 0
Start frame: 4956
End frame: 5476
"Identifier" 2.a.K
ID: 69
Schema: Sequence
Name: Isolation at Xanadu
Creator ID: 1
Representative: 5476
Children: 0
Start frame: 5476
End frame: 5654
"Identifier" 2.a.L
ID: 70
Schema: Sequence
Name: Death announced
Creator ID: 1
Representative: 5666
Children: 0
Start frame: 5654
End frame: 5741
"Identifier" 2.a.M
ID: 71
Schema: Shot
Name: Interview: Greatest film of all time
Creator ID: 3
Representative: 55332
Children: 0
Start frame: 55317
End frame: 55405
"Character" Robert Wise
ID: 447
292

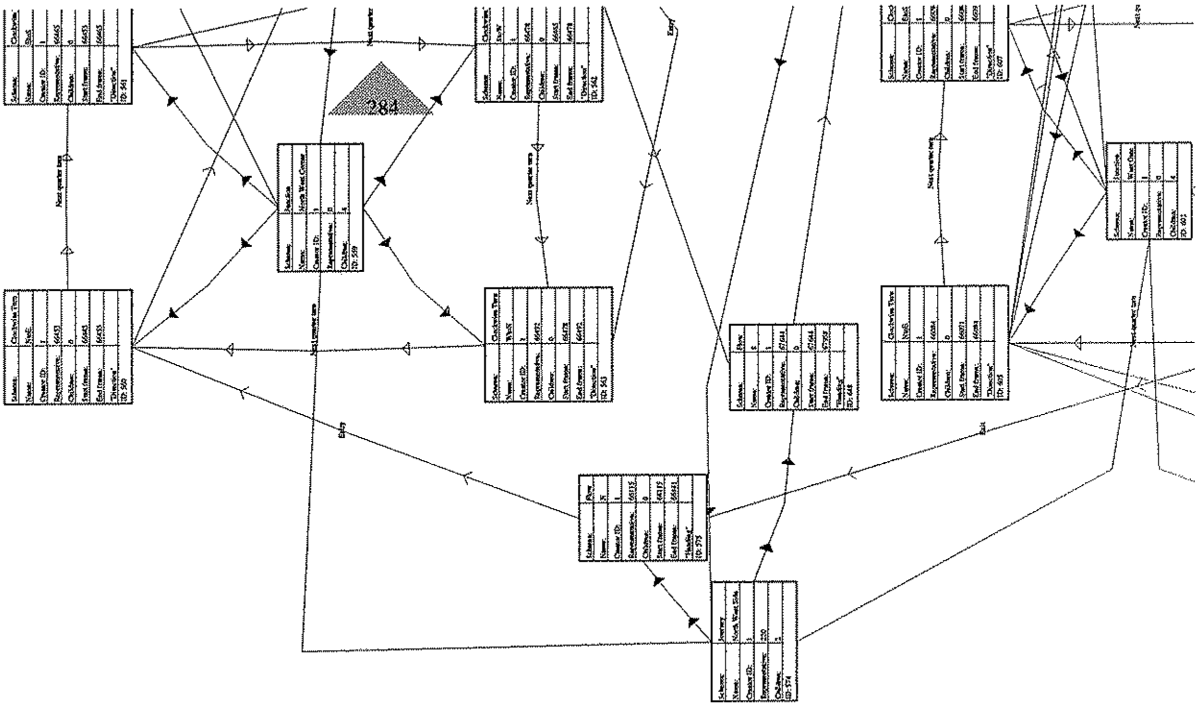

Schema: Movie | Name: Citizen Kane | Creator ID: 1 | Representative: 179 | Children: 18 | Start frame: 0 | End frame: 54574 | "Bibliography" Voted Greatest Film of All Time by the British Film Institute | ID: 48
Schema: Part | Name: Thatcher library | Creator ID: 1 | Representative: 7969 | Children: 3 | Start frame: 7953 | End frame: 13925 | "Identifier" 4 | ID: 103
Schema: Part | Name: Bernstein office | Creator ID: 1 | Representative: 13941 | Children: 3 | Start frame: 13925 | End frame: 22725 | "Identifier" 5 | ID: 134
Schema: Scene | Name: Thompson enters and reads Thatchers manuscript | Creator ID: 1 | Representative: 8555 | Children: 0 | Start frame: 7953 | End frame: 8720 | "Identifier" 4.a | "Approx date" | ID: 106
Schema: Sub-Part | Name: First flashback | Creator ID: 1 | Representative: 13644 | Children: 4 | Start frame: 8720 | End frame: 13082 | "Type" Flashback | ID: 123
Schema: Scene | Name: Thompson leaves library | Creator ID: 1 | Representative: 13749 | Children: 0 | Start frame: 13644 | End frame: 13925 | "Identifier" 4.f | "Approx date" | ID: 111
Schema: Scene | Name: Thompson visits Bernstein | Creator ID: 1 | Representative: 13941 | Children: 1 | Start frame: 13925 | End frame: 15181 | "Identifier" 5.a | "Approx date" | ID: 135
Schema: Sub-Part | Name: Second flashback | Creator ID: 1 | Representative: 15201 | Children: 5 | Start frame: 15181 | End frame: 22040 | "Type" Flashback | ID: 136
Schema: Scene | Name: Bernstein co | Creator ID: 1 | Representative: 22054 | Children: 0 | Start frame: 22040 | End frame: 22725 | "Identifier" 5.g | "Approx date" | ID: 137
Schema: Shot | Name: Thompson meets Bernstein | Creator ID: 1 | Representative: 13944 | Children: 0 | Start frame: 13925 | End frame: 15181 | "Character" Thompson | ID: 1449
Sequence | Invitation at Xanadu | 1 | 5476 | 0 | 5475 | 5656 | 2.a.L
Schema: Sequence | Name: Death announced | Creator ID: 1 | Representative: 5696 | Children: 0 | Start frame: 5654 | End frame: 5742 | "Identifier" 2.a.M | ID: 71
Schema: Scene | Name: Kanes mother sends the boy off with Thatcher | Creator ID: 1 | Representative: 9722 | Children: 2 | Start frame: 8720 | End frame: 10630 | "Identifier" 4.b | "Approx date" 1871 | ID: 107
Schema: Scene | Name: Kane grows up and buys the Inquirer | Creator ID: 1 | Representative: 11102 | Children: 0 | Start frame: 10630 | End frame: 11134 | "Identifier" 4.c | "Approx date" 1895 | ID: 108
Schema: Scene | Name: Kane launches the Inquirers attack on big business | Creator ID: 1 | Representative: 11259 | Children: 0 | Start frame: 11134 | End frame: 12536 | "Identifier" 4.d | "Approx date" 1897 | ID: 109
Schema: Scene | Name: The Depression: Kane sells Thatcher his newspaper chain | Creator ID: 1 | Representative: 12568 | Children: 0 | Start frame: 12536 | End frame: 13082 | "Identifier" 4.e | "Approx date" 1929 | ID: 110
Schema: Scene | Name: Kane takes over Inquirer | Creator ID: 1 | Representative: 15201 | Children: 2 | Start frame: 15181 | End frame: 18265 | "Identifier" 5.b | "Approx date" 1895 | ID: 138
Schema: Scene | Name: Montage: The Inquirers growth | Creator ID: 1 | Representative: 18290 | Children: 0 | Start frame: 18265 | End frame: 18578 | "Identifier" 5.c | "Approx date" 1900 | ID: 139
Schema: Scene | Name: Party: the Inquirer celebrates getting the Chronicle staff | Creator ID: 1 | Representative: 19451 | Children: 2 | Start frame: 18578 | End frame: 20790 | "Identifier" 5.d | "Approx date" 1906 | ID: 140
Schema: Scene | Name: Leland and Bernstein discuss Kanes trip abroad | Creator ID: 1 | Representative: 21065 | Children: 0 | Start frame: 20790 | End frame: 21297 | "Identifier" 5.e | "Approx date" 1909 | ID: 141
Schema: Scene | Name: Kane returns with is fianc | Creator ID: 1 | Representative: 21433 | Children: 1 | Start frame: 21297 | End frame: 22040 | "Identifier" 5.f | "Approx date" 1910 | ID: 142
Schema: Shot | Name: For the union flag | Creator ID: 1 | Representative: 8854 | Children: 0 | Start frame: 8798 | End frame: 9598 | "Character" Boy Charlie | ID: 451
Schema: Shot | Name: Mother calls Charles In, Thatcher tries to take him. | Creator ID: 1 | Representative: 9598 | Children: 0 | Start frame: 9598 | End frame: 10477 | "Character" Mother | ID: 1452
Schema: Shot | Name: Kane enters newspaper | Creator ID: 1 | Representative: 15474 | Children: 0 | Start frame: 15420 | End frame: 15519 | "Character" Kane and Tabedish | ID: 543
Schema: Shot | Name: Flustered Carter | Creator ID: 1 | Representative: 16967 | Children: 0 | Start frame: 16961 | End frame: 16984 | "Character" Carter | ID: 1465
Schema: Shot | Name: Kane photographs new staff | Creator ID: 1 | Representative: 18717 | Children: 0 | Start frame: 18578 | End frame: 18822 | "Character" Kane and chronicle staff | ID: 526
Schema: Sequence | Name: Kane dances song | Creator ID: 1 | Representative: 20586 | Children: 0 | Start frame: 19709 | End frame: 20789 | "Identifier" 5.d.1 | ID: 538
Schema: Shot | Name: Welcome home | Creator ID: 1 | Representative: 0 | Children: 0 | Start frame: 21297 | End frame: 21744 | "Character" Kan | ID: 449
278
293

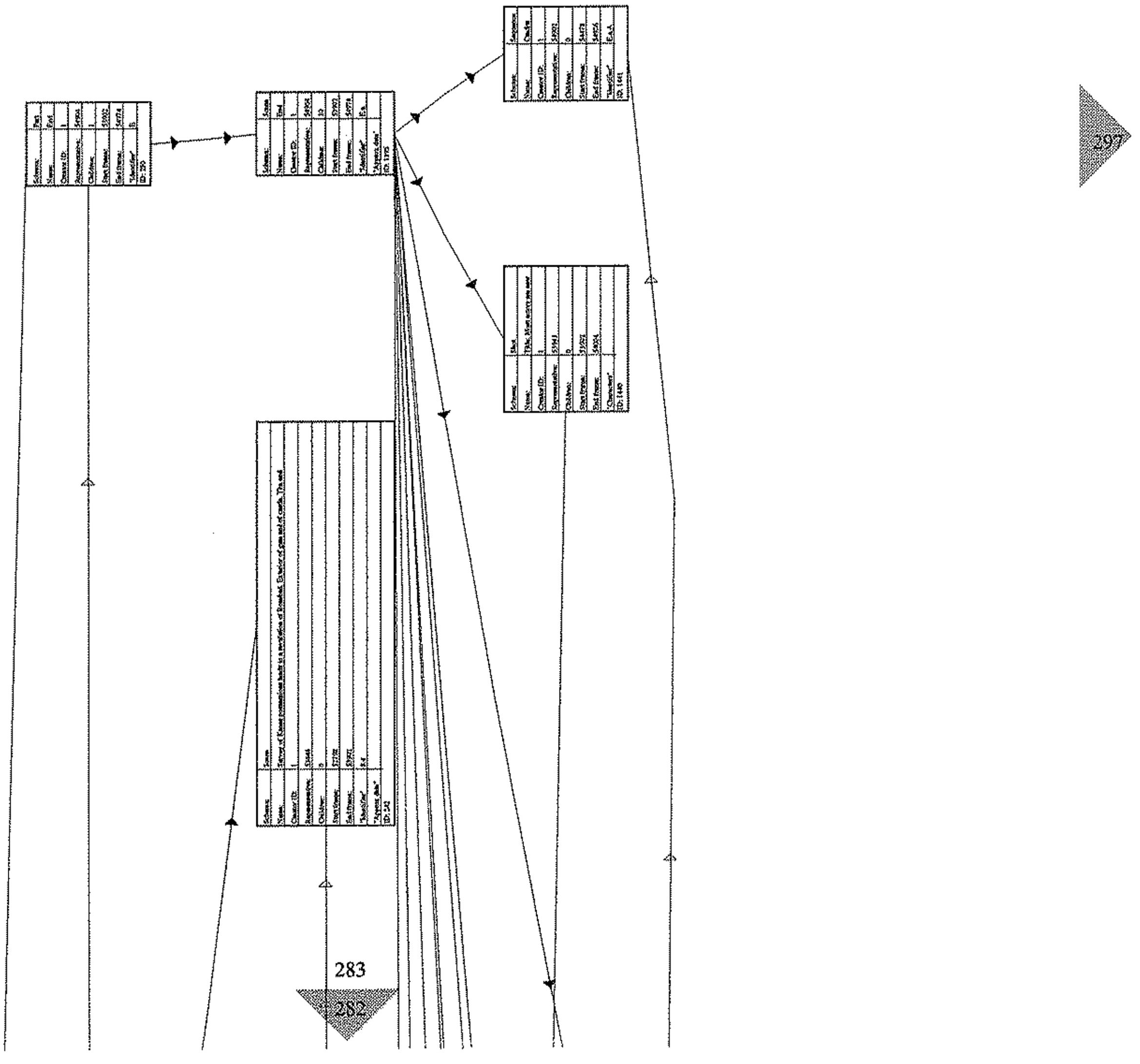


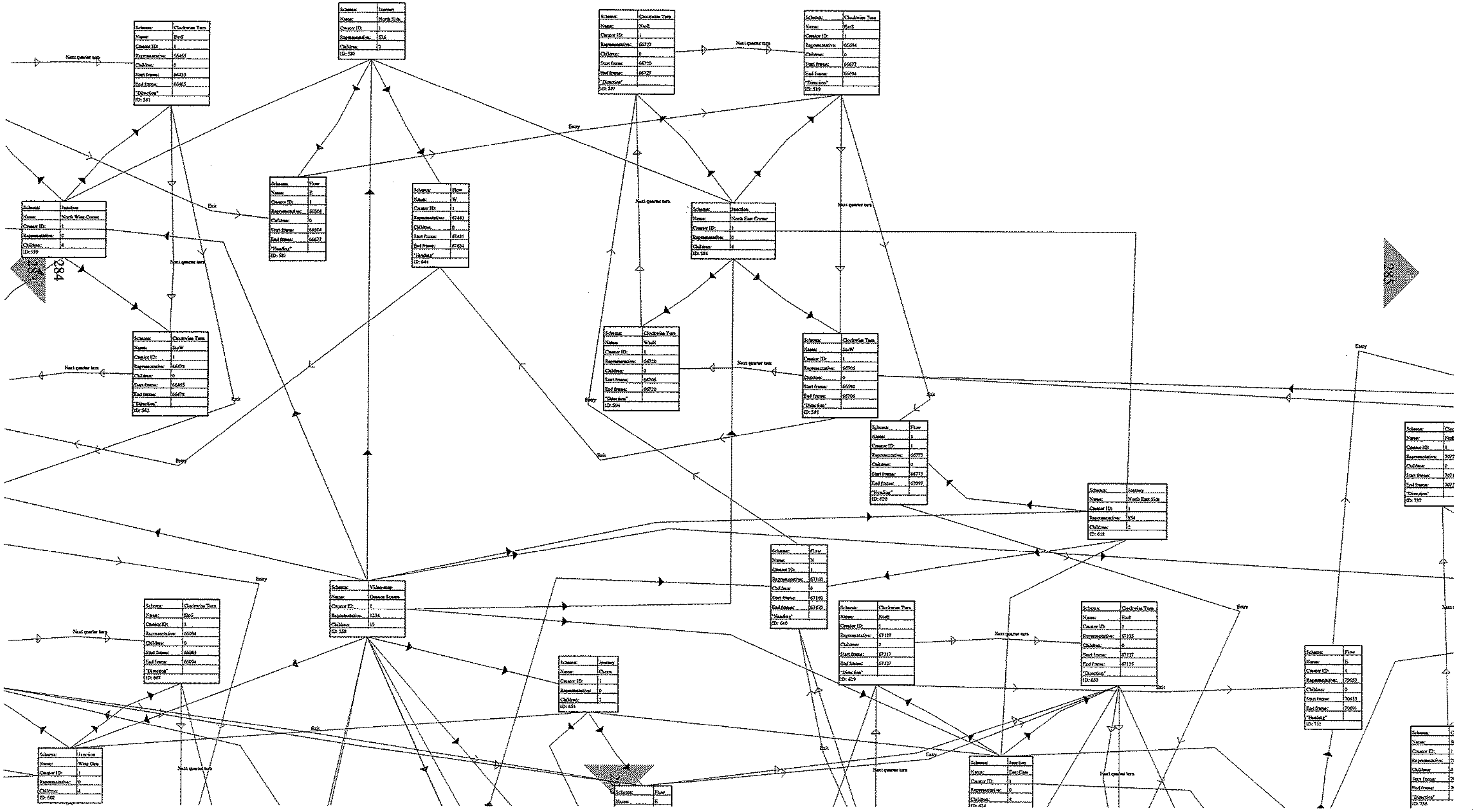
Schema: Part
Name: Nursing home
Creator ID: 1
Representative: 23156
Children: 5
Start frame: 22725
End frame: 39214
"Identifier" 6
ID: 154
Schema: Scene
Name: Bernstein concludes his reminiscences
Creator ID: 1
Representative: 22056
Children: 0
Start frame: 22040
End frame: 22725
"Identifier" 5.g
"Approx date"
ID: 157
Schema: Scene
Name: Thompson talks with Leland
Creator ID: 1
Representative: 23156
Children: 1
Start frame: 22725
End frame: 23895
"Identifier" 6.a
"Approx date"
ID: 155
Schema: Sub-Part
Name: Third flashback
Creator ID: 1
Representative: 24893
Children: 1
Start frame: 23895
End frame: 24893
"Type" Flashback
ID: 156
Schema: Scene
Name: Leland confesses his recollections
Creator ID: 1
Representative: 24893
Children: 0
Start frame: 24893
End frame: 25368
"Identifier" 6.c
"Approx date"
ID: 157
Schema: Sub-Part
Name: Third flashback (continued)
Creator ID: 1
Representative: 25618
Children: 7
Start frame: 25368
End frame: 35164
"Type" Flashback
ID: 158
and Bernstein discuss Kane's trip abroad
Schema: Scene
Name: Kane returns with his fiancée Emily
Creator ID: 1
Representative: 21433
Children: 1
Start frame: 21297
End frame: 22040
"Identifier" 5.f
"Approx date" 1910
ID: 142
Schema: Shot
Name: Leland starts recollection
Creator ID: 1
Representative: 22502
Children: 0
Start frame: 22108
End frame: 23835
"Character" Leland
ID: 1444
Schema: Scene
Name: Breakfast table montage: Kane's marriage deteriorates
Creator ID: 1
Representative: 24361
Children: 1
Start frame: 23895
End frame: 24893
"Identifier" 6.b
"Approx date" 1915
ID: 160
Schema: Scene
Name: Kane meets Susan and goes to her house
Creator ID: 1
Representative: 25618
Children: 0
Start frame: 25368
End frame: 27854
"Identifier" 6.d
"Approx date" 1919
ID: 161
Schema: Scene
Name: Kane political campaign culminates in his speech
Creator ID: 1
Representative: 28054
Children: 0
Start frame: 27854
End frame: 29458
"Identifier" 6.e
"Approx date" 1920
ID: 162
Schema: Scene
Name: Kane confronts Gettys, Emily, Susan
Creator ID: 1
Representative: 30022
Children: 1
Start frame: 29458
End frame: 32226
"Identifier" 6.f
"Approx date" 1920
ID: 163
Schema: Scene
Name: Kane loses election and Leland asks to be transferred
Creator ID: 1
Representative: 32276
Children: 0
Start frame: 32226
End frame: 34704
"Identifier" 6.g
"Approx date" 1921
ID: 164
Schema: Scene
Name: Kane marries Susan
Creator ID: 1
Representative: 34723
Children: 0
Start frame: 34704
End frame: 35010
"Identifier" 6.h
"Approx date" 1922
ID: 165
Schema: Scene
Name: Susan opens p
Creator ID: 1
Representative: 35034
Children: 0
Start frame: 35010
End frame: 35562
"Identifier" 6.i
"Approx date" 1923
ID: 166
Schema: Shot
Name: Welcome home Mr Kane
Creator ID: 1
Representative: 0
Children: 0
Start frame: 21297
End frame: 21744
"Character" Kan
ID: 443
Schema: Shot
Name: Sometimes I would prefer a rival of flesh and blood
Creator ID: 1
Representative: 24394
Children: 0
Start frame: 24386
End frame: 24403
"Character" Emily
ID: 1457
Schema: Shot
Name: Kane decides to stay with Susan
Creator ID: 1
Representative: 31774
Children: 0
Start frame: 31641
End frame: 31994
"Character" Kan
ID: 1461
Schema: Scene
Name: Use of angles and spatial relationships
Schema: Scene
Name: Use of deep focus
Schema: Scene
Name: Lighting techniques
279
281
294

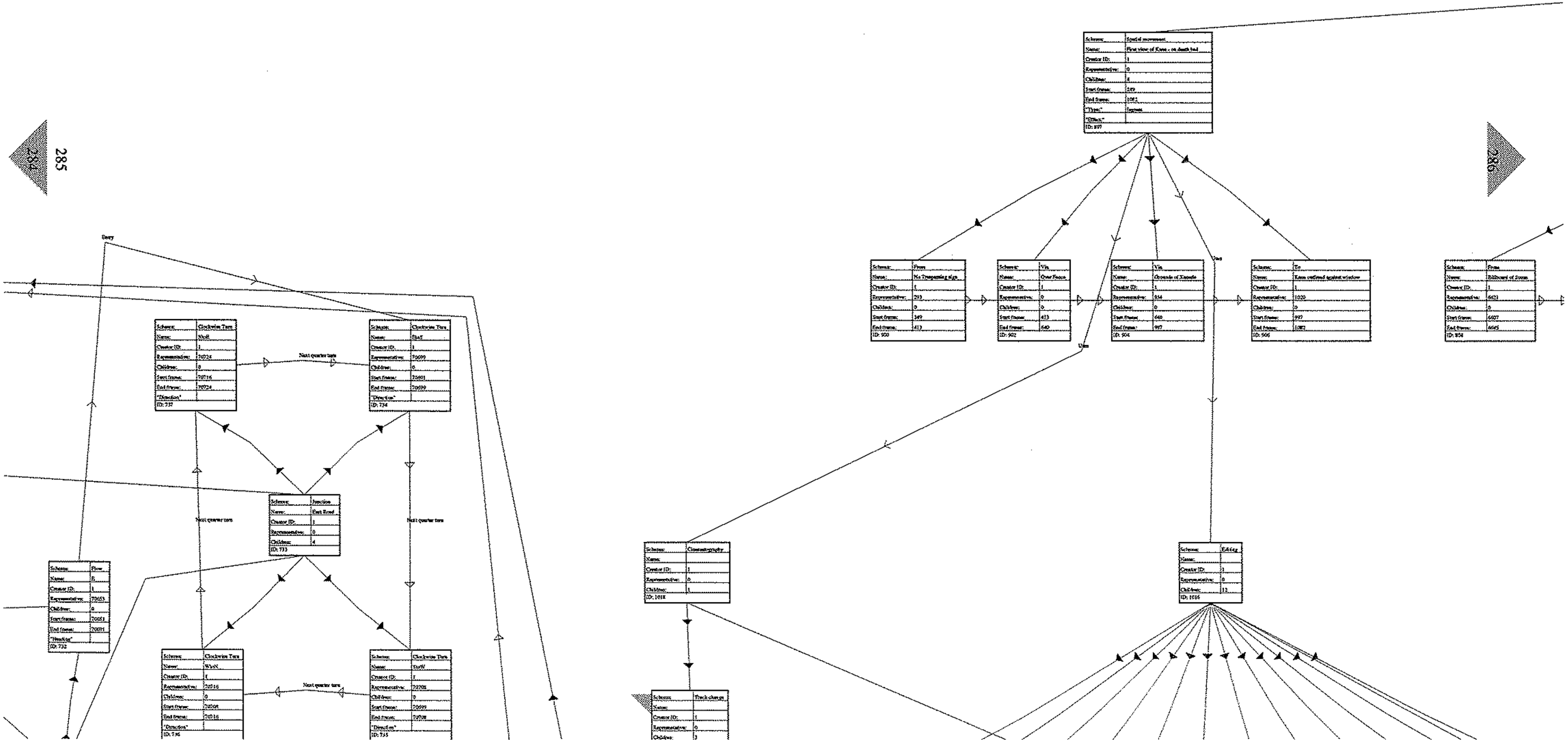

Schema: Part
Name: El Rancho nightclub
Creator ID: 1
Representative: 39271
Children: 5
Start frame: 39214
End frame: 49296
"Identifier" 7
ID: 192
Schema: Sub-Part
Name: Fourth flashback
Creator ID: 1
Representative: 40521
Children: 19
Start frame: 39865
End frame: 48885
"Type" Flashback
ID: 194
Part
Flashback (continued)
Schema: Scene
Name: Leland concludes his reminiscence
Creator ID: 1
Representative: 38209
Children: 0
Start frame: 38164
End frame: 39214
"Identifier" 6.k
"Approx date"
ID: 159
Schema: Scene
Name: Thompson talks with Susan
Creator ID: 1
Representative: 39410
Children: 1
Start frame: 39214
End frame: 39865
"Identifier" 7.a
"Approx date"
ID: 193
Schema: Text
Name: Susan starts her recollection
Creator ID: 1
Representative: 39858
Children: 0
Start frame: 39343
End frame: 39865
"Character" Susan
ID: 1448
Schema: Scene
Name: Kane meeting Susan
Creator ID: 1
Representative: 34725
Children: 0
Start frame: 34706
End frame: 35030
"Identifier" 5.h
"Approx date" 1922
ID: 165
Schema: Scene
Name: Susan opera premiere
Creator ID: 1
Representative: 35034
Children: 0
Start frame: 35030
End frame: 35662
"Identifier" 6.i
"Approx date" 1923
ID: 166
Schema: Scene
Name: Reunion; Leland is drunk, Kane finishes Leland's review
Creator ID: 1
Representative: 36493
Children: 0
Start frame: 35662
End frame: 38164
"Identifier" 6.j
"Approx date" 1923
ID: 167
Schema: Scene
Name: Susan rehearses her singing
Creator ID: 1
Representative: 40048
Children: 0
Start frame: 39865
End frame: 41003
"Identifier" 7.b
"Approx date" 1922
ID: 195
Schema: Scene
Name: Susan opera premiere
Creator ID: 1
Representative: 41025
Children: 0
Start frame: 41003
End frame: 42624
"Identifier" 7.c
"Approx date" 1923
ID: 196
Schema: Scene
Name: Kane insists that Susan goes on singing
Creator ID: 1
Representative: 42594
Children: 0
Start frame: 42624
End frame: 43570
"Identifier" 7.d
"Approx date" 1923
ID: 197
Schema: Scene
Name: Montage: Susan's opera career
Creator ID: 1
Representative: 43730
Children: 0
Start frame: 43570
End frame: 43895
"Identifier" 7.e
"Approx date" 1924
ID: 198
Schema: Scene
Name: Susan attempts suicide and Kane promises she can quit singing
Creator ID: 1
Representative: 44222
Children: 0
Start frame: 43895
End frame: 45228
"Identifier" 7.f
"Approx date" 1925
ID: 199
Schema: Scene
Name: Xanadu: Susan bored
Creator ID: 1
Representative: 45350
Children: 0
Start frame: 45228
End frame: 46143
"Identifier" 7.g
"Approx date" 1926
ID: 200
Schema: Scene
Name: Montage: Susan plays with jigsaw puzzle
Creator ID: 1
Representative: 46154
Children: 0
Start frame: 46143
End frame: 46297
"Identifier" 7.h
"Approx date" 1926
ID: 201
Schema: Scene
Name: Xanadu: Kane promis
Creator ID: 1
Representative: 46693
Children: 0
Start frame: 46297
End frame: 46754
"Identifier" 7.i
"Approx date" 1927
ID: 202
282
295
Schema: Scene
Name: Lighting technique
Schema: Scene
Name: Use of models
Schema: Scene
Name: Soundtrack
Schema: Scene
Schema: Scene

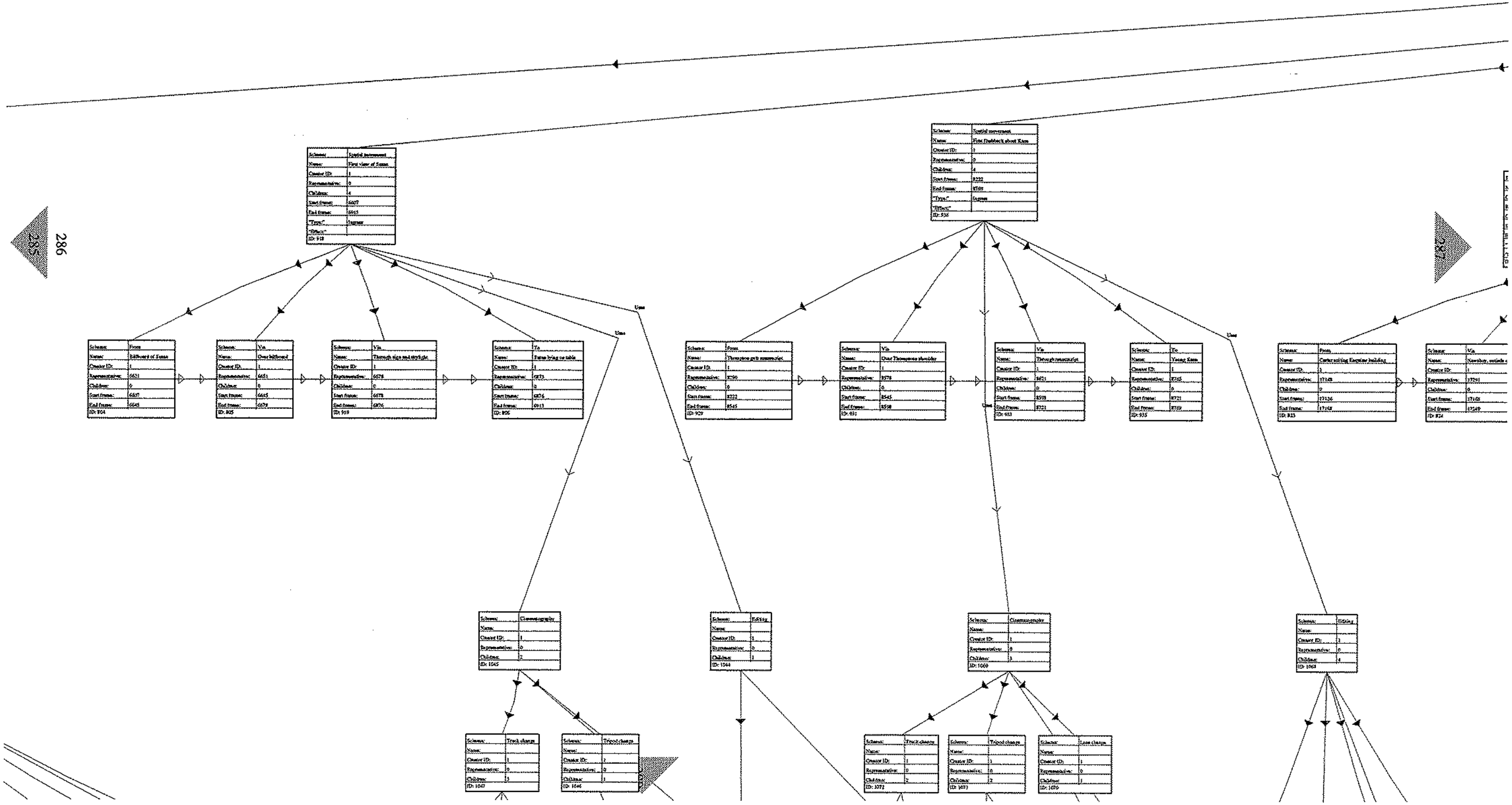

282
281
283
296
Schema: Part
Name: Xanadu
Creator ID: 1
Representative: 49515
Children: 4
Start frame: 49256
End frame: 53002
"Identifier" 8
ID: 234
Schema: Scene
Name: Susan concludes her reminiscence
Creator ID: 1
Representative: 49210
Children: 0
Start frame: 48885
End frame: 49256
"Identifier" 7.l
"Approx date"
ID: 205
Schema: Scene
Name: Thompson talks with Raymond
Creator ID: 1
Representative: 49331
Children: 0
Start frame: 49256
End frame: 49719
"Identifier" 8.a
"Approx date"
ID: 235
Schema: Sub-Part
Name: Fifth flashback
Creator ID: 1
Representative: 51180
Children: 1
Start frame: 49719
End frame: 51262
"Type" Flashback
ID: 236
Schema: Scene
Name: Raymond concludes his reminiscence, Thompson talks with the other reporters, all leave
Creator ID: 1
Representative: 51284
Children: 1
Start frame: 51262
End frame: 52798
"Identifier" 8.c
"Approx date"
ID: 240
Schema:
Name:
Creator ID:
Representative:
Children:
Start frame:
End frame:
"Identifier"
"Approx date"
ID: 241
Scene
Montage: Susan plays with jigsaw puzzle
1
46158
0
46143
46297
7.h
1926
Schema: Scene
Name: Xanadu: Kane promises picnic
Creator ID: 1
Representative: 46693
Children: 0
Start frame: 46297
End frame: 46754
"Identifier" 7.i
"Approx date" 1927
ID: 202
Schema: Scene
Name: Picnic: Kane slaps Susan
Creator ID: 1
Representative: 47163
Children: 0
Start frame: 46754
End frame: 47748
"Identifier" 7.j
"Approx date" 1927
ID: 203
Schema: Scene
Name: Xanadu: Susan leaves Kane
Creator ID: 1
Representative: 48056
Children: 0
Start frame: 47748
End frame: 48885
"Identifier" 7.k
"Approx date" 1928
ID: 204
Schema: Scene
Name: Kane destroys Susans room and picks up paperweight containing Rosebud
Creator ID: 1
Representative: 50666
Children: 1
Start frame: 49719
End frame: 51262
"Identifier" 8.b
"Approx date" 1929
ID: 238
Schema: Shot
Name: Raymond concludes
Creator ID: 1
Representative: 51517
Children: 0
Start frame: 51262
End frame: 51579
"Characters" Raymond
ID: 1453
Schema: Shot
Name: Kane walks through Xanadu alone.
Creator ID: 1
Representative: 51181
Children: 0
Start frame: 51102
End frame: 51262
"Characters" Old Kane
ID: 455

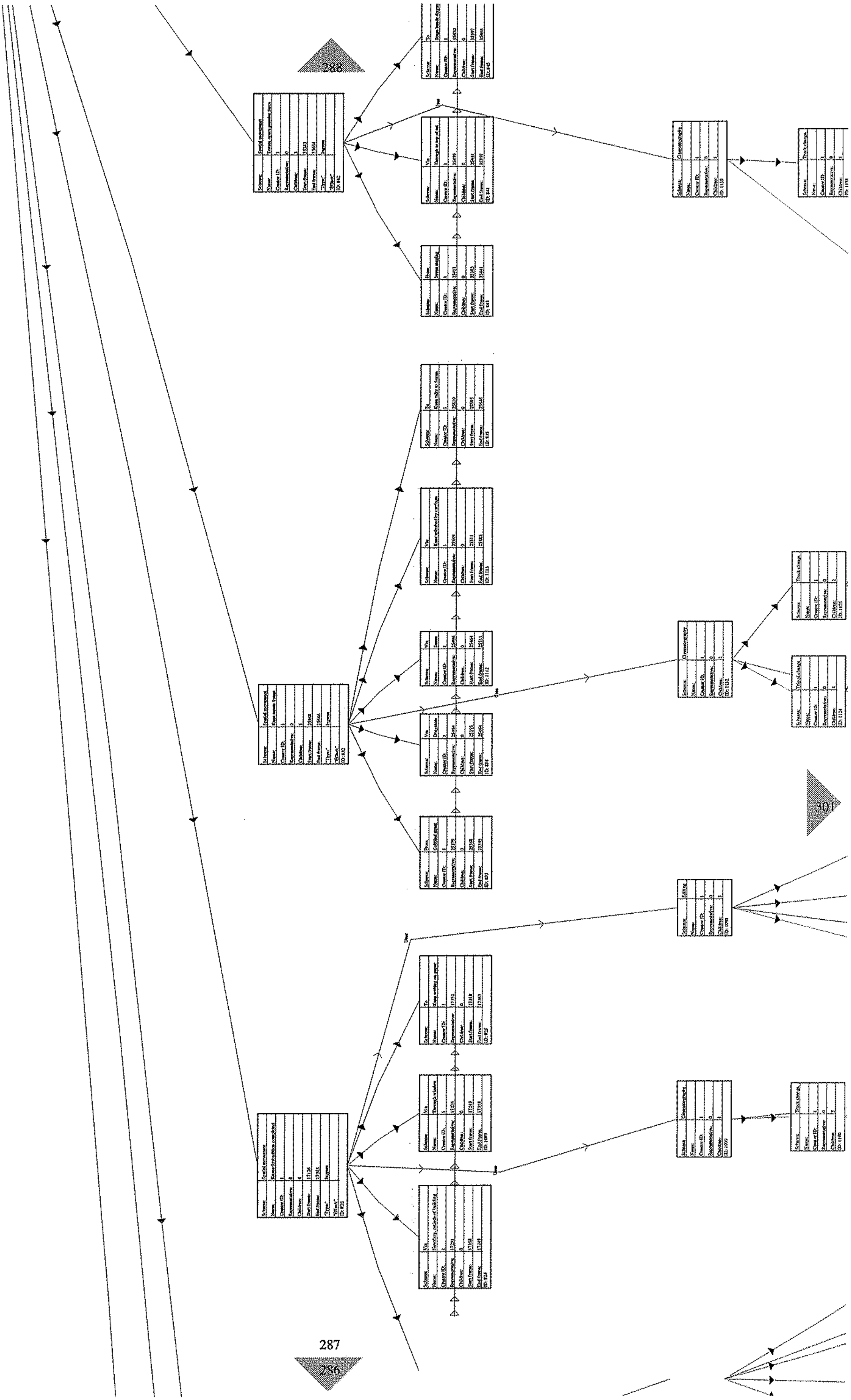

284
Next quarter turn
Next quarter turn
Next quarter turn
Entry
Exit
Schema: Direction
Name: North West Corner
ID: 559
Schema: Clockwise Turn
ID: 560
ID: 561
ID: 562
ID: 563
Schema: Journey
Name: North West Side
ID: 574
ID: 575

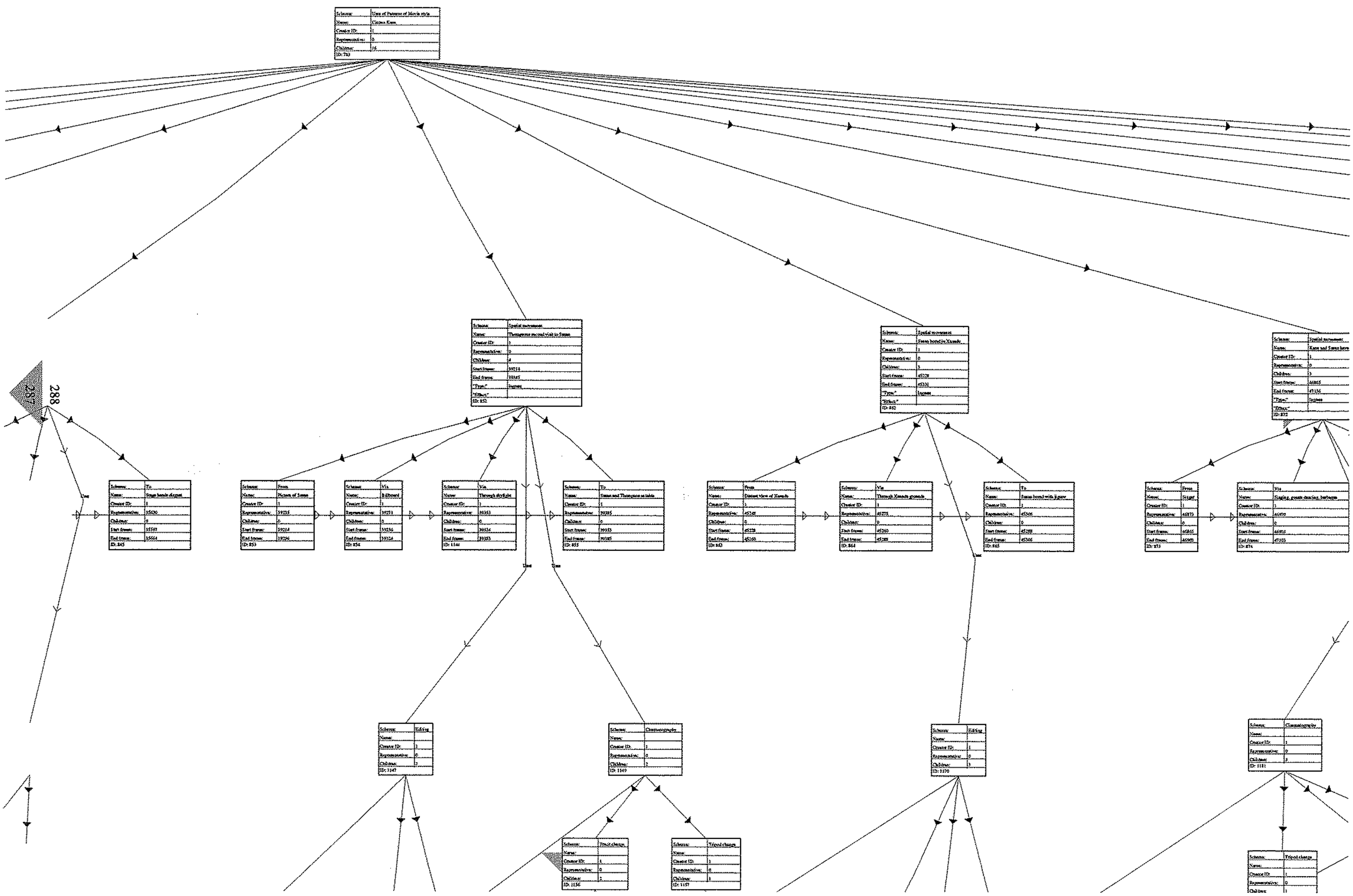

282
297


Schema: Clockwise Turn
Name: EtoS
Creator ID: 1
Representative: 66465
Children: 0
Start frame: 66453
End frame: 66465
"Direction"
ID: 561
Schema: Journey
Name: North Side
Creator ID: 1
Representative: 576
Children: 2
ID: 580
Schema: Clockwise Turn
Name: NtoE
Creator ID: 1
Representative: 66727
Children: 0
Start frame: 66720
End frame: 66727
"Direction"
ID: 597
Schema: Clockwise Turn
Name: EtoS
Creator ID: 1
Representative: 66694
Children: 0
Start frame: 66677
End frame: 66694
"Direction"
ID: 589
Schema: Junction
Name: North West Corner
Creator ID: 1
Representative: 0
Children: 4
ID: 559
Schema: Flow
Name: E
Creator ID: 1
Representative: 66604
Children: 0
Start frame: 66604
End frame: 66677
"Heading"
ID: 583
Schema: Flow
Name: W
Creator ID: 1
Representative: 67481
Children: 0
Start frame: 67451
End frame: 67534
"Heading"
ID: 644
Schema: Junction
Name: North East Corner
Creator ID: 1
Representative: 0
Children: 4
ID: 586
Schema: Clockwise Turn
Name: StoW
Creator ID: 1
Representative: 66478
Children: 0
Start frame: 66465
End frame: 66478
"Direction"
ID: 562
Schema: Clockwise Turn
Name: WtoN
Creator ID: 1
Representative: 66720
Children: 0
Start frame: 66706
End frame: 66720
"Direction"
ID: 594
Schema: Clockwise Turn
Name: StoW
Creator ID: 1
Representative: 66706
Children: 0
Start frame: 66694
End frame: 66706
"Direction"
ID: 591
Schema: Flow
Name: S
Creator ID: 1
Representative: 66773
Children: 0
Start frame: 66773
End frame: 67097
"Heading"
ID: 620
Schema: Journey
Name: North East Side
Creator ID: 1
Representative: 854
Children: 2
ID: 618
Schema: Video-map
Name: Queens Square
Creator ID: 1
Representative: 1234
Children: 15
ID: 558
Schema: Flow
Name: N
Creator ID: 1
Representative: 67160
Children: 0
Start frame: 67160
End frame: 67470
"Heading"
ID: 640
Schema: Clockwise Turn
Name: EtoS
Creator ID: 1
Representative: 66094
Children: 0
Start frame: 66084
End frame: 66094
"Direction"
ID: 607
Schema: Journey
Name: Cheers
Creator ID: 1
Representative: 0
Children: 2
ID: 654
Schema: Clockwise Turn
Name: NtoE
Creator ID: 1
Representative: 67127
Children: 0
Start frame: 67117
End frame: 67127
"Direction"
ID: 629
Schema: Clockwise Turn
Name: EtoS
Creator ID: 1
Representative: 67135
Children: 0
Start frame: 67127
End frame: 67135
"Direction"
ID: 630
Schema: Flow
Name: E
Creator ID: 1
Representative: 70653
Children: 0
Start frame: 70653
End frame: 70691
"Heading"
ID: 732
Schema: Junction
Name: West Gate
Creator ID: 1
Representative: 0
Children: 4
ID: 602
Schema: Junction
Name: East Gate
Creator ID: 1
Representative: 0
Children: 4
ID: 624
Schema: Flow
Name: E
ID: 737
ID: 736
Next quarter turn
Entry
Exit
283
284
285

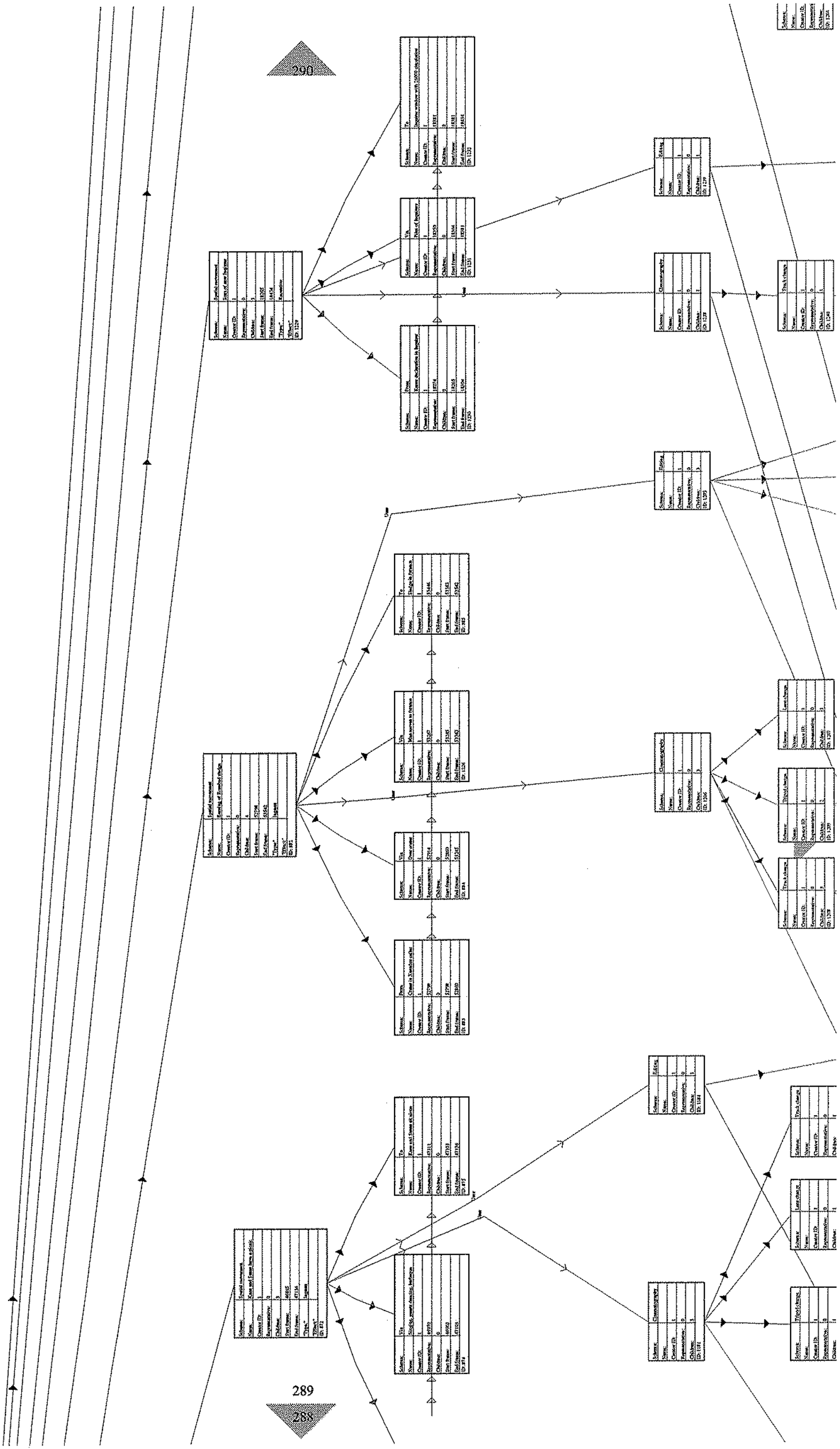

Schema: Spatial movement
Name: First view of Kane - on death bed
Creator ID: 1
Representative: 0
Children: 5
Start frame: 349
End frame: 1082
"Type:" Inpoint
"Effect:"
ID: 897
Schema: From
Name: No Trespassing sign
Creator ID: 1
Representative: 293
Children: 0
Start frame: 349
End frame: 413
ID: 900
Schema: Via
Name: Over Fence
Creator ID: 1
Representative: 0
Children: 0
Start frame: 413
End frame: 640
ID: 902
Schema: Via
Name: Grounds of Xanadu
Creator ID: 1
Representative: 934
Children: 0
Start frame: 640
End frame: 997
ID: 904
Schema: To
Name: Kane outlined against window
Creator ID: 1
Representative: 1020
Children: 0
Start frame: 997
End frame: 1082
ID: 906
Schema: From
Name: Billboard of Susan
Creator ID: 1
Representative: 6621
Children: 0
Start frame: 6607
End frame: 6645
ID: 858
Uses
Uses
Schema: Cinematography
Name:
Creator ID: 1
Representative: 0
Children: 1
ID: 1018
Schema: Editing
Name:
Creator ID: 1
Representative: 0
Children: 12
ID: 1016
Schema: Track change
Name:
Creator ID: 1
Representative: 0
Children: 3
Story
Schema: Clockwise Turn
Name: NtoE
Creator ID: 1
Representative: 70724
Children: 0
Start frame: 70716
End frame: 70724
"Direction"
ID: 737
Next quarter turn
Schema: Clockwise Turn
Name: EtoS
Creator ID: 1
Representative: 70699
Children: 0
Start frame: 70691
End frame: 70699
"Direction"
ID: 734
Schema: Junction
Name: East Road
Creator ID: 1
Representative: 0
Children: 4
ID: 733
Next quarter turn
Next quarter turn
Schema: Flow
Name: f5
Creator ID: 1
Representative: 70653
Children: 0
Start frame: 70651
End frame: 70691
"Heading"
ID: 732
Schema: Clockwise Turn
Name: WtoN
Creator ID: 1
Representative: 70716
Children: 0
Start frame: 70708
End frame: 70716
"Direction"
ID: 736
Next quarter turn
Schema: Clockwise Turn
Name: StoW
Creator ID: 1
Representative: 70708
Children: 0
Start frame: 70699
End frame: 70708
"Direction"
ID: 735
284
286


285
287
Schema: Spatial movement
Name: First view of Susan
Creator ID: 1
Representative: 0
Children: 4
Start frame: 6607
End frame: 6915
"Type": Ingress
"Effect":
ID: 918
Schema: From
Name: Billboard of Susan
Creator ID: 1
Representative: 6631
Children: 0
Start frame: 6607
End frame: 6645
ID: 804
Schema: Via
Name: Over billboard
Creator ID: 1
Representative: 6651
Children: 0
Start frame: 6645
End frame: 6678
ID: 805
Schema: Via
Name: Through sign and skylight
Creator ID: 1
Representative: 6678
Children: 0
Start frame: 6678
End frame: 6876
ID: 919
Schema: To
Name: Susan lying on table
Creator ID: 1
Representative: 6873
Children: 0
Start frame: 6876
End frame: 6913
ID: 806
Uses
Uses
Schema: Spatial movement
Name: First flashback about Kane
Creator ID: 1
Representative: 0
Children: 4
Start frame: 8222
End frame: 8769
"Type": Ingress
"Effect":
ID: 938
Schema: From
Name: Thompson gets manuscript
Creator ID: 1
Representative: 8250
Children: 0
Start frame: 8222
End frame: 8545
ID: 929
Schema: Via
Name: Over Thompson's shoulder
Creator ID: 1
Representative: 8578
Children: 0
Start frame: 8545
End frame: 8598
ID: 931
Schema: Via
Name: Through manuscript
Creator ID: 1
Representative: 8621
Children: 0
Start frame: 8598
End frame: 8721
ID: 933
Schema: To
Name: Young Kane
Creator ID: 1
Representative: 8765
Children: 0
Start frame: 8721
End frame: 8769
ID: 935
Uses
Uses
Schema: From
Name: Carter exiting Enquirer building
Creator ID: 1
Representative: 17148
Children: 0
Start frame: 17136
End frame: 17165
ID: 823
Schema: Via
Name: Newsboy, outside c
Creator ID: 1
Representative: 17291
Children: 0
Start frame: 17165
End frame: 17249
ID: 824
Schema: Cinematography
Name:
Creator ID: 1
Representative: 0
Children: 2
ID: 1045
Schema: Editing
Name:
Creator ID: 1
Representative: 0
Children: 1
ID: 1044
Schema: Cinematography
Name:
Creator ID: 1
Representative: 0
Children: 3
ID: 1069
Schema: Editing
Name:
Creator ID: 1
Representative: 0
Children: 4
ID: 1063
Schema: Track change
Name:
Creator ID: 1
Representative: 0
Children: 3
ID: 1047
Schema: Tripod change
Name:
Creator ID: 2
Representative: 0
Children: 1
ID: 1046
Schema: Track change
Name:
Creator ID: 1
Representative: 0
Children: 2
ID: 1072
Schema: Tripod change
Name:
Creator ID: 1
Representative: 0
Children: 2
ID: 1073
Schema: Lens change
Name:
Creator ID: 1
Representative: 0
Children: 1
ID: 1076

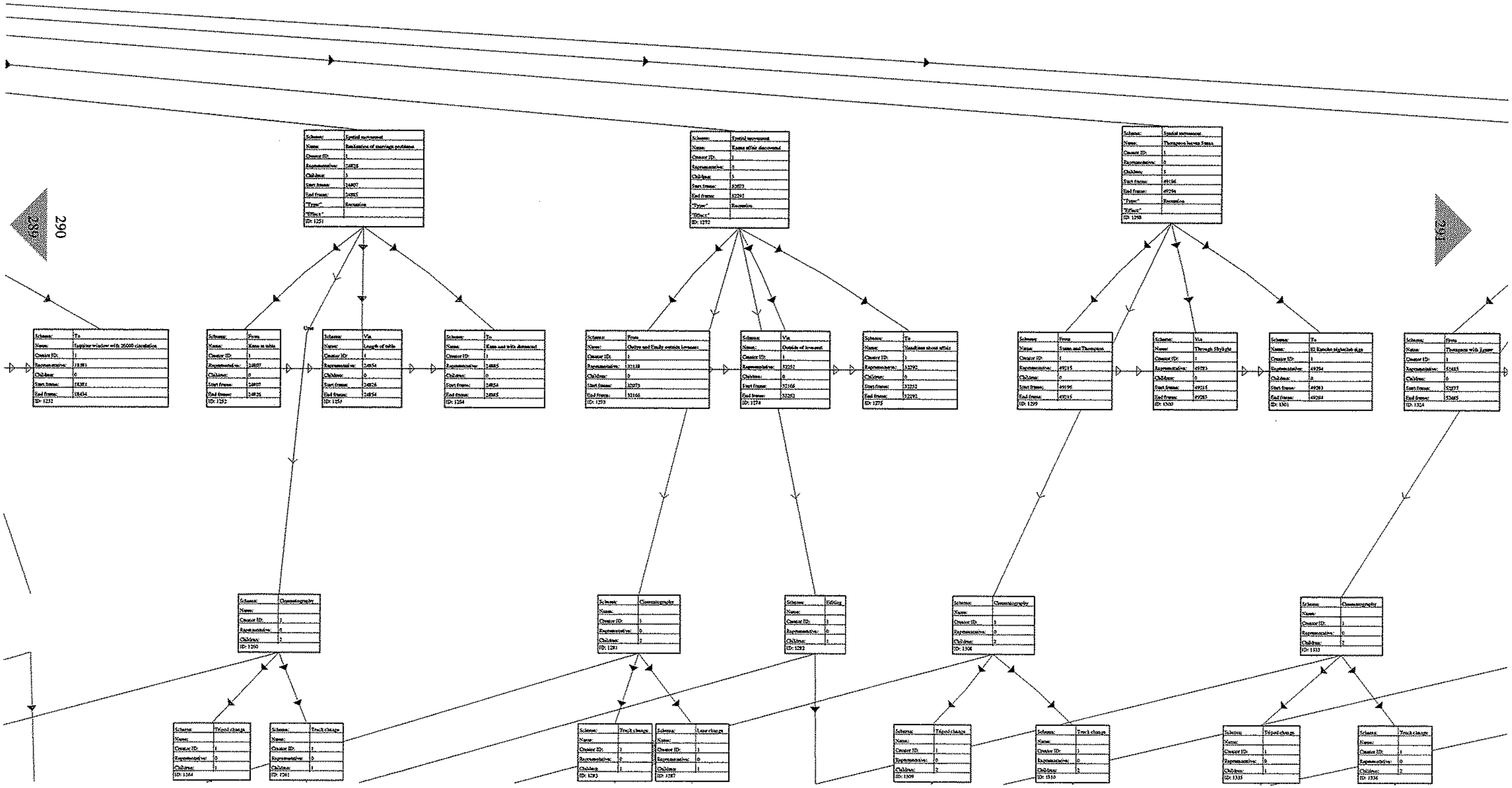

288
301
286
Schema:
Name:
Creator ID:
Representative:
Children:
Start frame:
End frame:
"Type"
"Effect"
Spatial movement
Via
Pov
To
Cobbled street
Chronotopography
Editing
Truck change
Uses

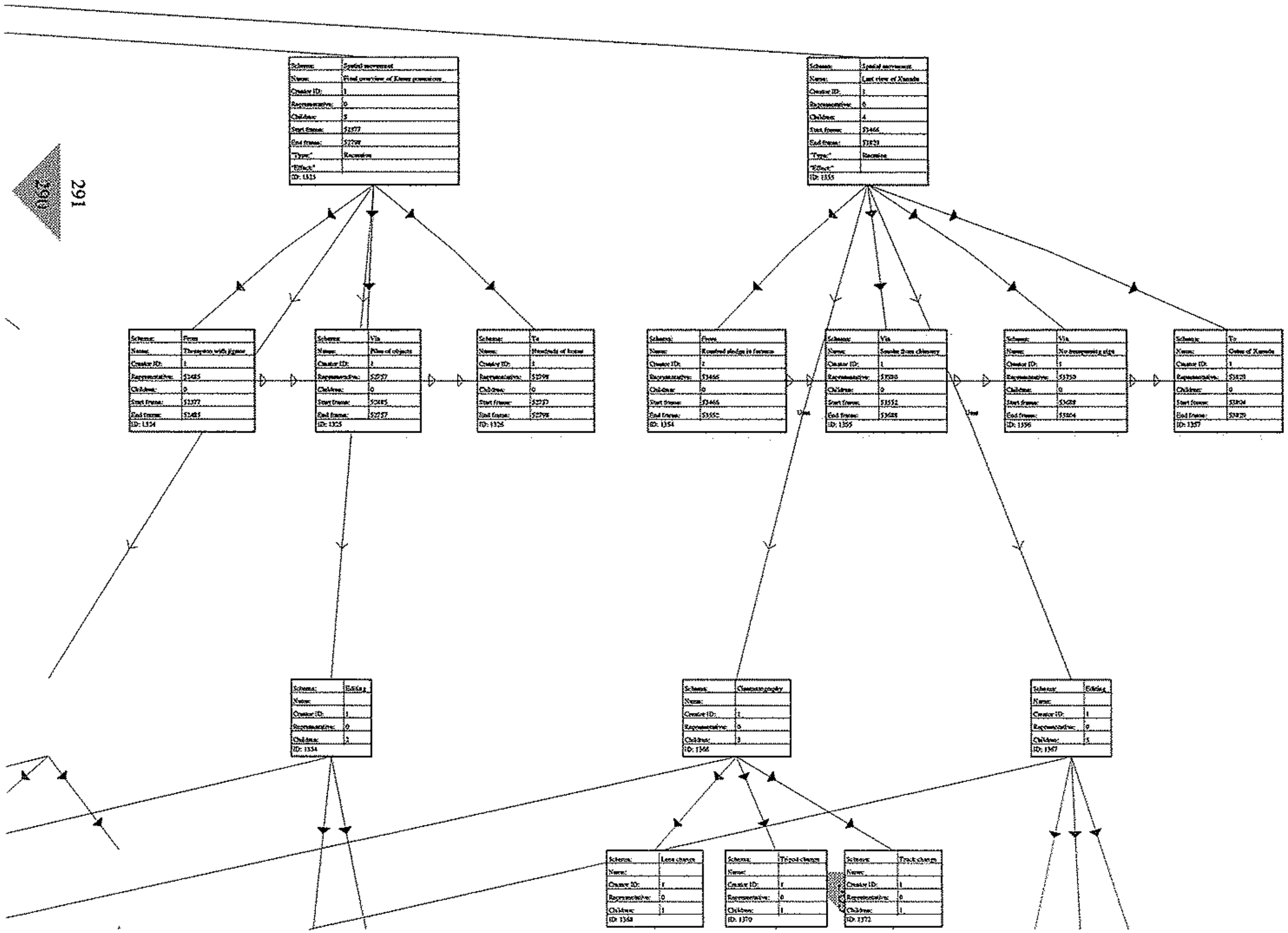

Schema: Uses of Patterns of Movie style
Name: Citizen Kane
Creator ID: 1
Representative: 0
Children: 16
ID: 783
Schema: Spatial movement
Name: Thompsons second visit to Susan
Creator ID: 1
Representative: 0
Children: 4
Start frame: 39214
End frame: 39385
"Type:" Ingress
"Effect:"
ID: 852
Schema: Spatial movement
Name: Susan bored in Xanadu
Creator ID: 1
Representative: 0
Children: 3
Start frame: 45228
End frame: 45331
"Type:" Ingress
"Effect:"
ID: 862
Schema: Spatial movement
Name: Kane and Susan have
Creator ID: 1
Representative: 0
Children: 3
Start frame: 46865
End frame: 47136
"Type:" Ingress
"Effect:"
ID: 872
288
287
Use
Schema: To
Name: Stage hands disgust
Creator ID: 1
Representative: 35620
Children: 0
Start frame: 35597
End frame: 35664
ID: 845
Schema: From
Name: Picture of Susan
Creator ID: 1
Representative: 39235
Children: 0
Start frame: 39214
End frame: 39256
ID: 853
Schema: Via
Name: Billboard
Creator ID: 1
Representative: 39271
Children: 0
Start frame: 39256
End frame: 39324
ID: 854
Schema: Via
Name: Through skylight
Creator ID: 1
Representative: 39353
Children: 0
Start frame: 39324
End frame: 39353
ID: 1144
Schema: To
Name: Susan and Thompson at table
Creator ID: 1
Representative: 39385
Children: 0
Start frame: 39353
End frame: 39385
ID: 855
Use
Use
Schema: From
Name: Distant view of Xanadu
Creator ID: 1
Representative: 45245
Children: 0
Start frame: 45228
End frame: 45260
ID: 863
Schema: Via
Name: Through Xanadu grounds
Creator ID: 1
Representative: 45275
Children: 0
Start frame: 45260
End frame: 45288
ID: 864
Schema: To
Name: Susan bored with jigsaw
Creator ID: 1
Representative: 45306
Children: 0
Start frame: 45288
End frame: 45306
ID: 865
Use
Schema: From
Name: Singer
Creator ID: 1
Representative: 46873
Children: 0
Start frame: 46865
End frame: 46903
ID: 873
Schema: Via
Name: Singing, guests dancing, barbecue
Creator ID: 1
Representative: 46970
Children: 0
Start frame: 46903
End frame: 47103
ID: 874
Schema: Editing
Name:
Creator ID: 1
Representative: 0
Children: 2
ID: 1147
Schema: Cinematography
Name:
Creator ID: 1
Representative: 0
Children: 2
ID: 1149
Schema: Track change
Name:
Creator ID: 1
Representative: 0
Children: 2
ID: 1156
Schema: Tripod change
Name:
Creator ID: 1
Representative: 0
Children: 1
ID: 1157
Schema: Editing
Name:
Creator ID: 1
Representative: 0
Children: 3
ID: 1170
Schema: Cinematography
Name:
Creator ID: 1
Representative: 0
Children: 3
ID: 1181
Schema: Tripod change
Name:
Creator ID: 1
Representative: 0
Children: 1

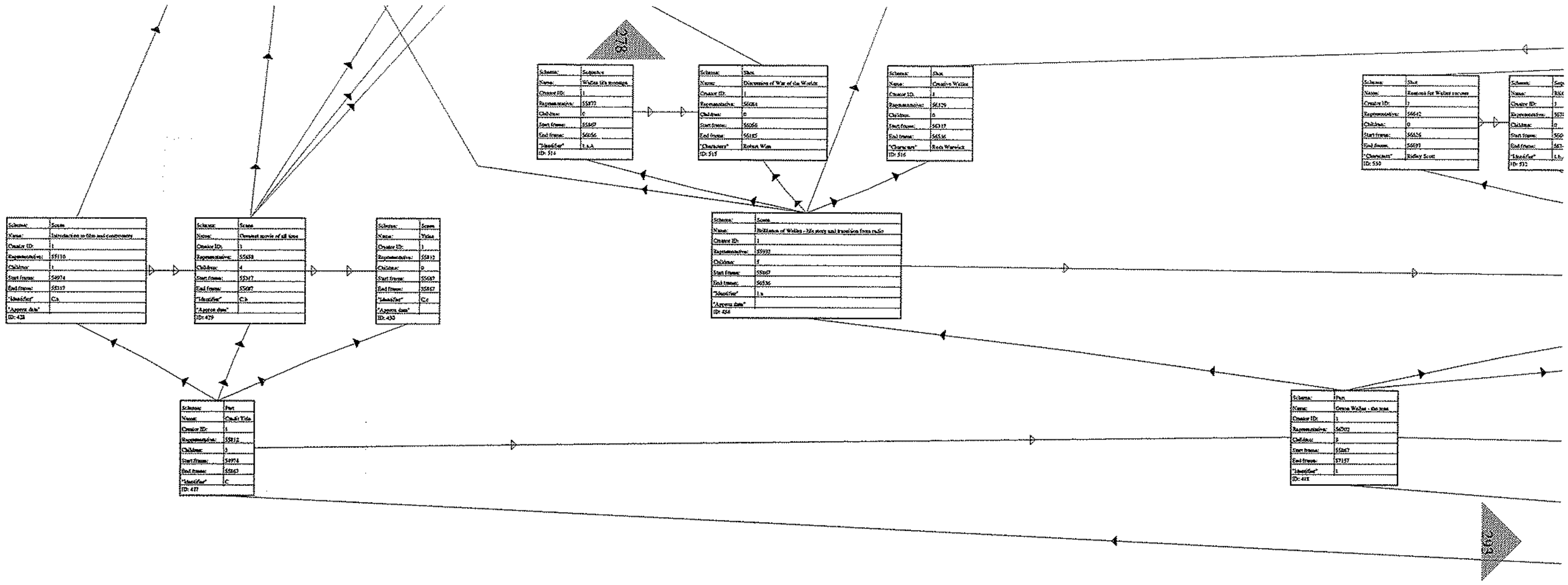
290

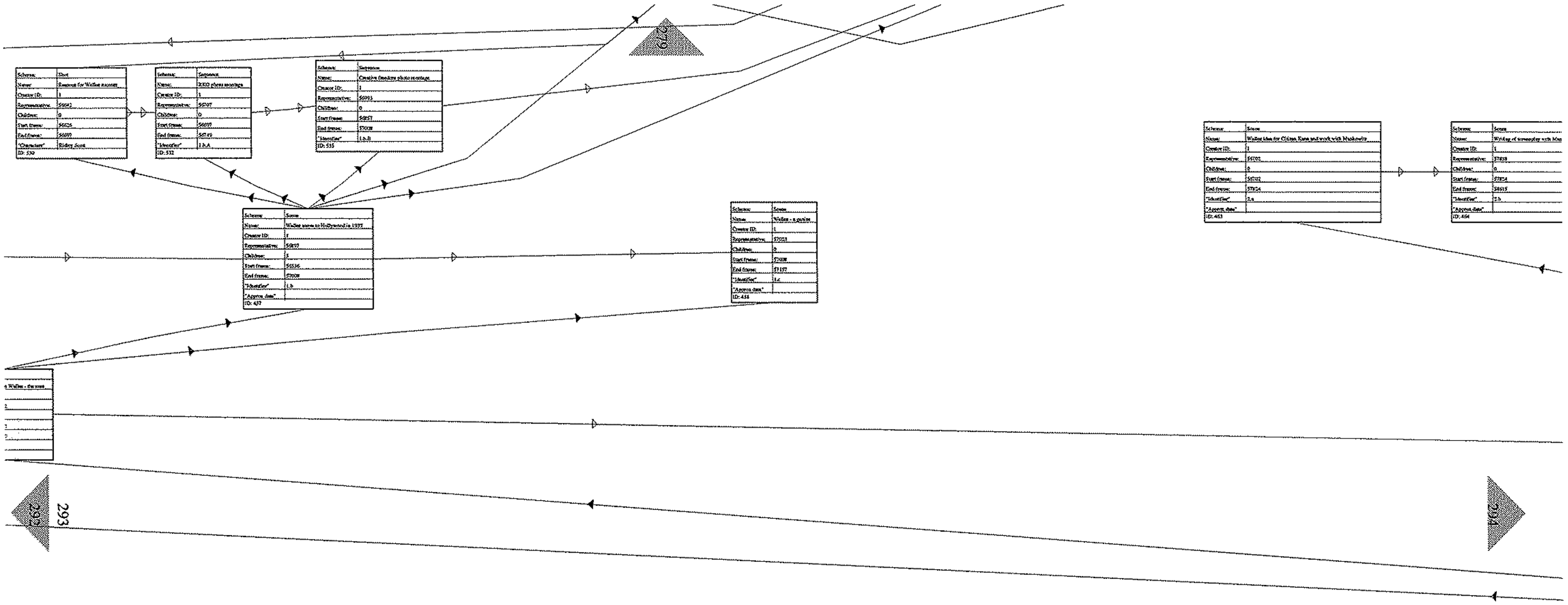

Schema:
Name:
Creator ID:
Representative:
Children:
Start frame:
End frame:
"Type"
"Effect"
Spatial movement
Via
To
From
User
Editing
Cinematography
Track change
Lens change
Tripod change
ID: 892
ID: 1236
ID: 884

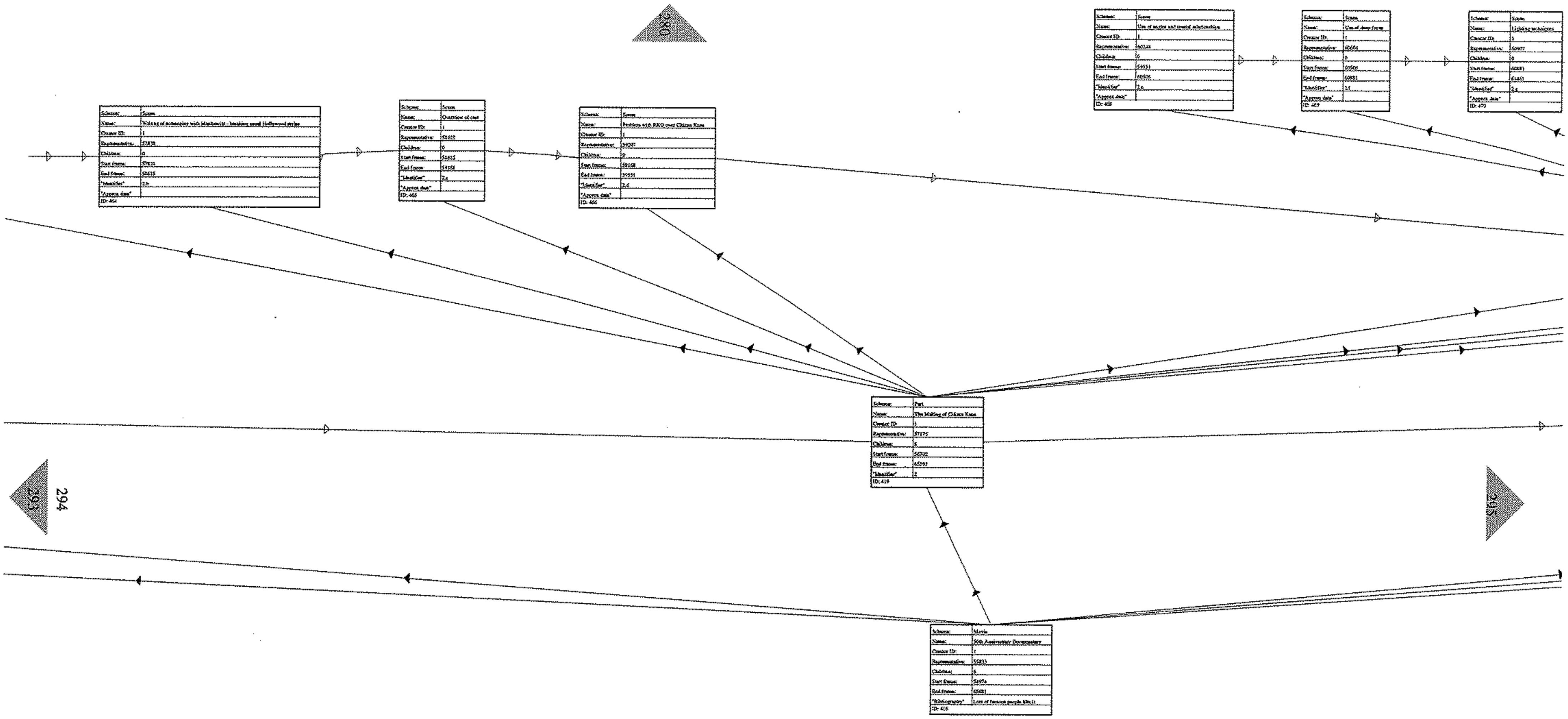
288

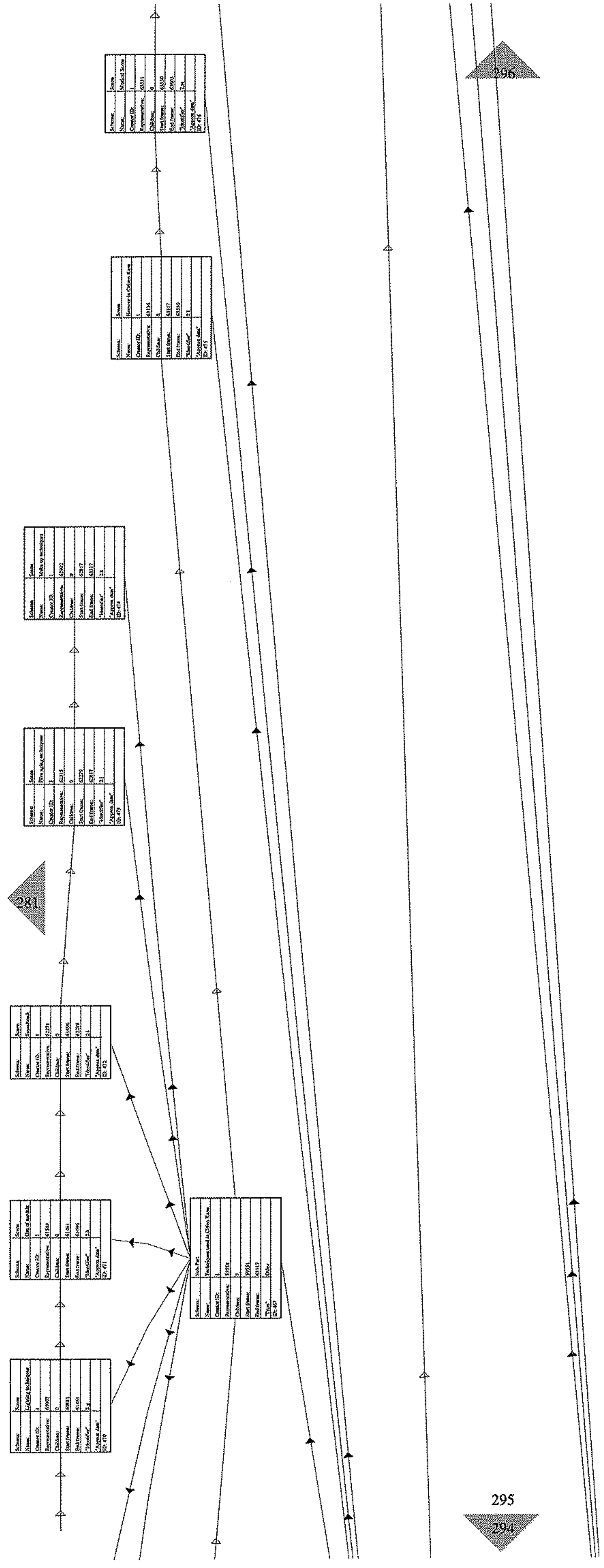

289
290
291
Schema: Spatial movement
Name: Realization of marriage problems
Creator ID: 1
Representative: 24825
Children: 3
Start frame: 24807
End frame: 24885
"Type": Recursion
"Effect"
ID: 1251
Schema: Spatial movement
Name: Kane affair discovered
Creator ID: 1
Representative: 0
Children: 3
Start frame: 32073
End frame: 32292
"Type": Recursion
"Effect"
ID: 1272
Schema: Spatial movement
Name: Thompson leaves Susan
Creator ID: 1
Representative: 0
Children: 3
Start frame: 49196
End frame: 49294
"Type": Recursion
"Effect"
ID: 1298
Schema: To
Name: Inquirer window with 26000 circulation
Creator ID: 1
Representative: 18383
Children: 0
Start frame: 18381
End frame: 18434
ID: 1232
Schema: From
Name: Kane at table
Creator ID: 1
Representative: 24807
Children: 0
Start frame: 24807
End frame: 24825
ID: 1252
Use
Schema: Via
Name: Length of table
Creator ID: 1
Representative: 24854
Children: 0
Start frame: 24826
End frame: 24854
ID: 1253
Schema: To
Name: Kane and wife distanced
Creator ID: 1
Representative: 24885
Children: 0
Start frame: 24854
End frame: 24885
ID: 1254
Schema: From
Name: Gettys and Emily outside lovenest
Creator ID: 1
Representative: 32138
Children: 0
Start frame: 32073
End frame: 32166
ID: 1273
Schema: Via
Name: Outside of lovenest
Creator ID: 1
Representative: 32252
Children: 0
Start frame: 32166
End frame: 32252
ID: 1274
Schema: To
Name: Headlines about affair
Creator ID: 1
Representative: 32292
Children: 0
Start frame: 32252
End frame: 32292
ID: 1275
Schema: From
Name: Susan and Thompson
Creator ID: 1
Representative: 49215
Children: 0
Start frame: 49196
End frame: 49215
ID: 1299
Schema: Via
Name: Through Skylight
Creator ID: 1
Representative: 49283
Children: 0
Start frame: 49215
End frame: 49283
ID: 1300
Schema: To
Name: El Rancho nightclub sign
Creator ID: 1
Representative: 49294
Children: 0
Start frame: 49283
End frame: 49294
ID: 1301
Schema: From
Name: Thompson with Raymond
Creator ID: 1
Representative: 53425
Children: 0
Start frame: 52977
End frame: 53485
ID: 1324
Schema: Cinematography
Name:
Creator ID: 3
Representative: 0
Children: 2
ID: 1250
Schema: Cinematography
Name:
Creator ID: 3
Representative: 0
Children: 2
ID: 1281
Schema: Editing
Name:
Creator ID: 3
Representative: 0
Children: 1
ID: 1282
Schema: Cinematography
Name:
Creator ID: 3
Representative: 0
Children: 2
ID: 1308
Schema: Cinematography
Name:
Creator ID: 3
Representative: 0
Children: 2
ID: 1333
Schema: Tripod change
Name:
Creator ID: 1
Representative: 0
Children: 1
ID: 1264
Schema: Track change
Name:
Creator ID: 3
Representative: 0
Children: 1
ID: 1261
Schema: Track change
Name:
Creator ID: 3
Representative: 0
Children: 1
ID: 1283
Schema: Lens change
Name:
Creator ID: 3
Representative: 0
Children: 1
ID: 1287
Schema: Tripod change
Name:
Creator ID: 1
Representative: 0
Children: 2
ID: 1309
Schema: Track change
Name:
Creator ID: 3
Representative: 0
Children: 2
ID: 1310
Schema: Tripod change
Name:
Creator ID: 1
Representative: 0
Children: 1
ID: 1335
Schema: Track change
Name:
Creator ID: 1
Representative: 0
Children: 2
ID: 1336

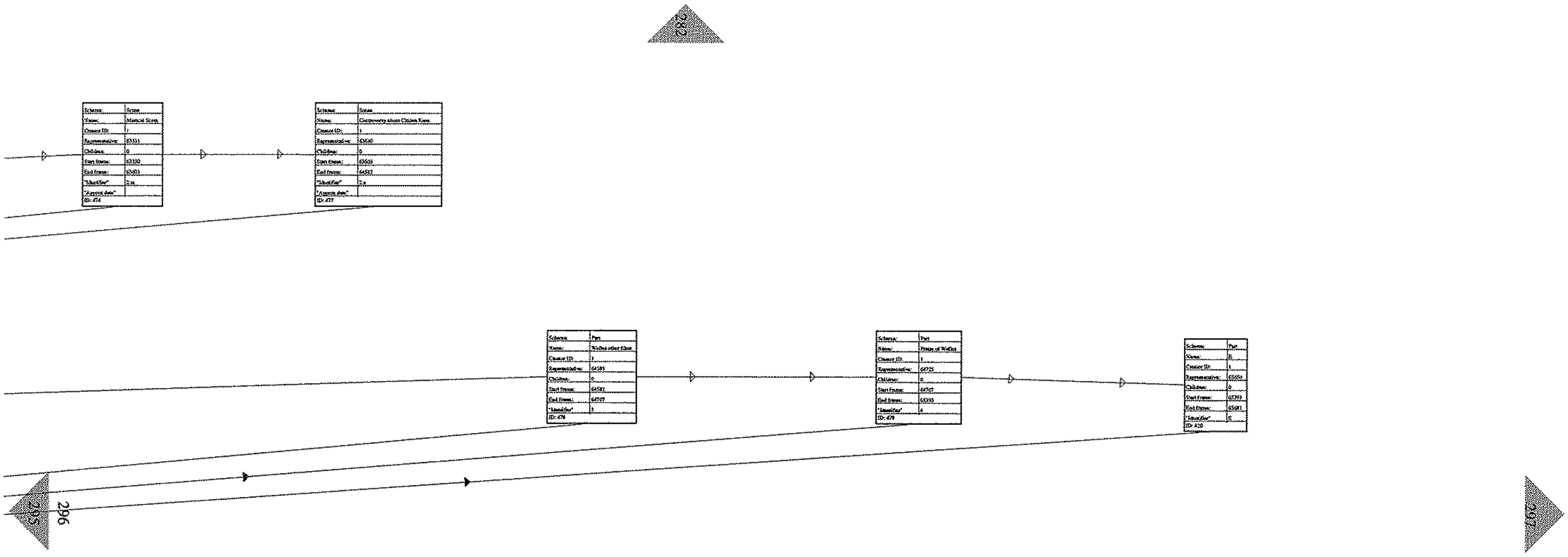

Schema: Spatial movement
Name: Final overview of Kane's possessions
Creator ID: 1
Representative: 0
Children: 3
Start frame: 52377
End frame: 52798
"Type": Recursion
"Effect"
ID: 1323
Schema: Spatial movement
Name: Last view of Xanadu
Creator ID: 1
Representative: 0
Children: 4
Start frame: 53466
End frame: 53829
"Type": Recursion
"Effect"
ID: 1353
Schema: From
Name: Thompson with jigsaw
Creator ID: 1
Representative: 52485
Children: 0
Start frame: 52377
End frame: 52485
ID: 1324
Schema: Via
Name: Piles of objects
Creator ID: 1
Representative: 52757
Children: 0
Start frame: 52485
End frame: 52757
ID: 1325
Schema: To
Name: Hundreds of boxes
Creator ID: 1
Representative: 52798
Children: 0
Start frame: 52757
End frame: 52798
ID: 1326
Schema: From
Name: Rosebud sledge in furnace
Creator ID: 2
Representative: 53466
Children: 0
Start frame: 53466
End frame: 53552
ID: 1354
Schema: Via
Name: Smoke from chimney
Creator ID: 1
Representative: 53520
Children: 0
Start frame: 53552
End frame: 53628
ID: 1355
Schema: Via
Name: No trespassing sign
Creator ID: 1
Representative: 53750
Children: 0
Start frame: 53628
End frame: 53804
ID: 1356
Schema: To
Name: Gates of Xanadu
Creator ID: 1
Representative: 53829
Children: 0
Start frame: 53804
End frame: 53829
ID: 1357
Uses
Uses
Schema: Editing
Name:
Creator ID: 1
Representative: 0
Children: 2
ID: 1334
Schema: Cinematography
Name:
Creator ID: 1
Representative: 0
Children: 3
ID: 1366
Schema: Editing
Name:
Creator ID: 1
Representative: 0
Children: 5
ID: 1367
Schema: Lens change
Name:
Creator ID: 1
Representative: 0
Children: 1
ID: 1368
Schema: Tripod change
Name:
Creator ID: 1
Representative: 0
Children: 1
ID: 1370
Schema: Track change
Name:
Creator ID: 1
Representative: 0
Children: 1
ID: 1372


278
Schema: | Sequence
Name: | Welles life montage
Creator ID: | 1
Representative: | 55877
Children: | 0
Start frame: | 55467
End frame: | 56056
"Identifier" | 1.a.A
ID: 514
Schema: | Shot
Name: | Discussion of War of the Worlds
Creator ID: | 1
Representative: | 56084
Children: | 0
Start frame: | 56056
End frame: | 56185
"Characters" | Robert Wise
ID: 515
Schema: | Shot
Name: | Creative Welles
Creator ID: | 1
Representative: | 56129
Children: | 0
Start frame: | 56317
End frame: | 56556
"Characters" | Robert Warwick
ID: 516
Schema: | Shot
Name: | Reasons for Welles success
Creator ID: | 1
Representative: | 56642
Children: | 0
Start frame: | 56626
End frame: | 56693
"Characters" | Ridley Scott
ID: 530
Schema: | Seq
Name: | RKO
Creator ID: | 1
Representative: | 557
Children: | 0
Start frame: | 560
End frame: | 563
"Identifier" | 1.b.
ID: 532
Schema: | Scene
Name: | Introduction to film and controversy
Creator ID: | 1
Representative: | 55110
Children: | 1
Start frame: | 54974
End frame: | 55317
"Identifier" | C.a
"Approx date"
ID: 428
Schema: | Scene
Name: | Greatest movie of all time
Creator ID: | 1
Representative: | 55658
Children: | 4
Start frame: | 55317
End frame: | 55687
"Identifier" | C.b
"Approx date"
ID: 429
Schema: | Scene
Name: | Titles
Creator ID: | 1
Representative: | 55812
Children: | 0
Start frame: | 55687
End frame: | 55867
"Identifier" | C.c
"Approx date"
ID: 430
Schema: | Scene
Name: | Brilliance of Welles - life story and transition from radio
Creator ID: | 1
Representative: | 55932
Children: | 5
Start frame: | 55867
End frame: | 56536
"Identifier" | 1.a
"Approx date"
ID: 456
Schema: | Part
Name: | Credit Title
Creator ID: | 1
Representative: | 55812
Children: | 3
Start frame: | 54974
End frame: | 55863
"Identifier" | C
ID: 427
Schema: | Part
Name: | Orson Welles - the man
Creator ID: | 1
Representative: | 56002
Children: | 3
Start frame: | 55867
End frame: | 57157
"Identifier" | 1
ID: 418
293

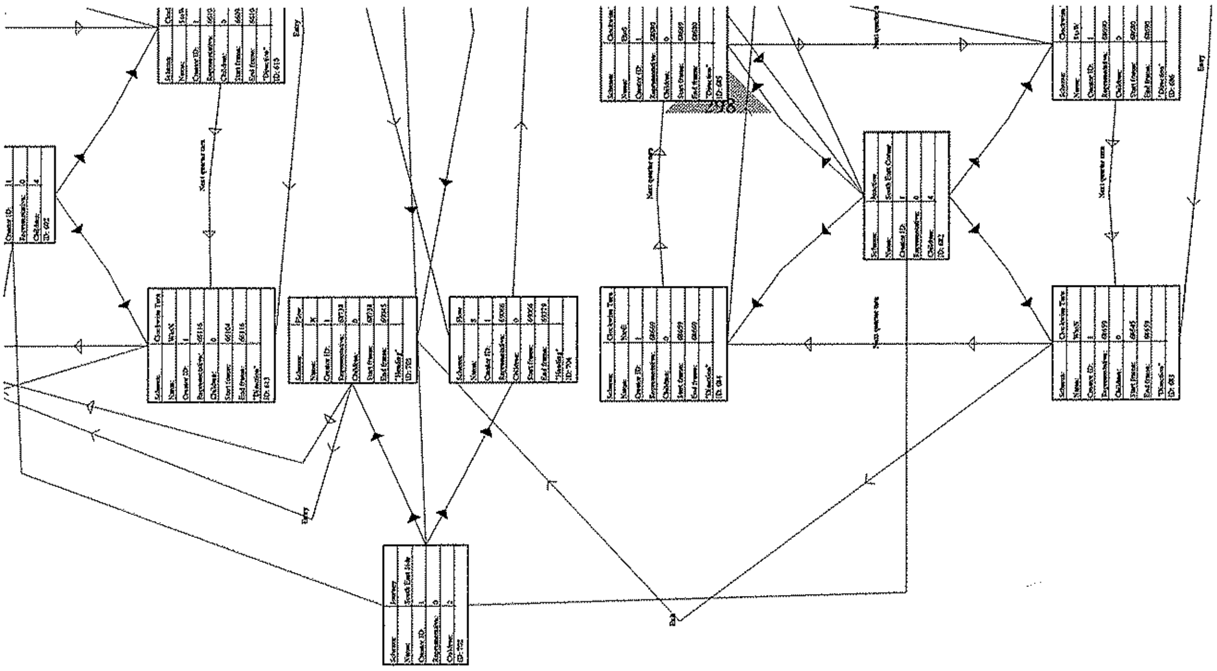

279
Scheme: Shot
Name: Response for Welles success
Creator ID: 1
Representative: 56642
Children: 0
Start frame: 56626
End frame: 56697
"Character" Ridley Scott
ID: 530
Scheme: Sequence
Name: RKO photo montage
Creator ID: 1
Representative: 56707
Children: 0
Start frame: 56697
End frame: 56849
"Identifier" 1.b.A
ID: 532
Scheme: Sequence
Name: Creative freedom photo montage
Creator ID: 1
Representative: 56933
Children: 0
Start frame: 56857
End frame: 57008
"Identifier" 1.b.B
ID: 535
Scheme: Scene
Name: Welles move to Hollywood in 1939
Creator ID: 1
Representative: 56997
Children: 3
Start frame: 56536
End frame: 57008
"Identifier" 1.b
"Approx date"
ID: 437
Scheme: Scene
Name: Welles - a genius
Creator ID: 1
Representative: 57053
Children: 0
Start frame: 57008
End frame: 57157
"Identifier" 1.c
"Approx date"
ID: 438
Scheme: Scene
Name: Welles idea for Citizen Kane and work with Mankowitz
Creator ID: 1
Representative: 57702
Children: 2
Start frame: 57702
End frame: 57854
"Identifier" 2.a
"Approx date"
ID: 453
Scheme: Scene
Name: Writing of screenplay with Man
Creator ID: 1
Representative: 57853
Children: 0
Start frame: 57854
End frame: 58615
"Identifier" 2.b
"Approx date"
ID: 454
n Welles - the man
2
3
7
292
293
294


280
Schema: Scene
Name: Writing of screenplay with Mankiewicz - breaking usual Hollywood styles
Creator ID: 1
Representative: 57838
Children: 0
Start frame: 57824
End frame: 58615
"Identifier" 2.b
"Appears date"
ID: 462
Schema: Scene
Name: Overview of cast
Creator ID: 1
Representative: 58622
Children: 0
Start frame: 58615
End frame: 59158
"Identifier" 2.c
"Appears date"
ID: 465
Schema: Scene
Name: Problems with RKO over Citizen Kane
Creator ID: 1
Representative: 59287
Children: 0
Start frame: 59168
End frame: 59551
"Identifier" 2.d
"Appears date"
ID: 466
Schema: Scene
Name: Use of angles and spatial relationships
Creator ID: 1
Representative: 60244
Children: 0
Start frame: 59551
End frame: 60506
"Identifier" 2.e
"Appears date"
ID: 468
Schema: Scene
Name: Use of deep focus
Creator ID: 1
Representative: 60604
Children: 0
Start frame: 60506
End frame: 60883
"Identifier" 2.f
"Appears date"
ID: 469
Schema: Scene
Name: Lighting techniques
Creator ID: 1
Representative: 60907
Children: 0
Start frame: 60883
End frame: 61461
"Identifier" 2.g
"Appears date"
ID: 470
Schema: Part
Name: The Making of Citizen Kane
Creator ID: 3
Representative: 57175
Children: 8
Start frame: 56702
End frame: 65393
"Identifier" 2
ID: 419
293
295
Schema: Movie
Name: 50th Anniversary Documentary
Creator ID: 1
Representative: 55833
Children: 6
Start frame: 54974
End frame: 65681
"Bibliography" Lots of famous people like it
ID: 416


296
Schema: Scene
Name: Medical Scene
Creator ID: 1
Representative: 63331
Children: 0
Start Frame: 63330
End Frame: 65005
"Identifier" 24
"Approx date"
ID: 476
Schema: Scene
Name: Humour in Citizen Kane
Creator ID: 1
Representative: 63135
Children: 0
Start Frame: 63117
End Frame: 63330
"Identifier" 23
"Approx date"
ID: 475
Schema: Scene
Name: Make-up technique
Creator ID: 1
Representative: 63092
Children: 0
Start Frame: 62817
End Frame: 63117
"Identifier" 23
"Approx date"
ID: 474
Schema: Scene
Name: Fine aging technique
Creator ID: 1
Representative: 62315
Children: 0
Start Frame: 62279
End Frame: 62817
"Identifier" 21
"Approx date"
ID: 473
281
Schema: Scene
Name: Soundtrack
Creator ID: 1
Representative: 62271
Children: 0
Start Frame: 61096
End Frame: 62279
"Identifier" 21
"Approx date"
ID: 472
Schema: Scene
Name: Use of models
Creator ID: 1
Representative: 61543
Children: 0
Start Frame: 61053
End Frame: 61096
"Identifier" 23
"Approx date"
ID: 471
Schema: Scene
Name: Lighting techniques
Creator ID: 1
Representative: 60937
Children: 0
Start Frame: 60853
End Frame: 61053
"Identifier" 23
"Approx date"
ID: 470
Schema: Sub-Part
Name: Techniques used in Citizen Kane
Creator ID: 1
Representative: 59958
Children: 7
Start Frame: 59551
End Frame: 63117
"Type" Other
ID: 467
295
294

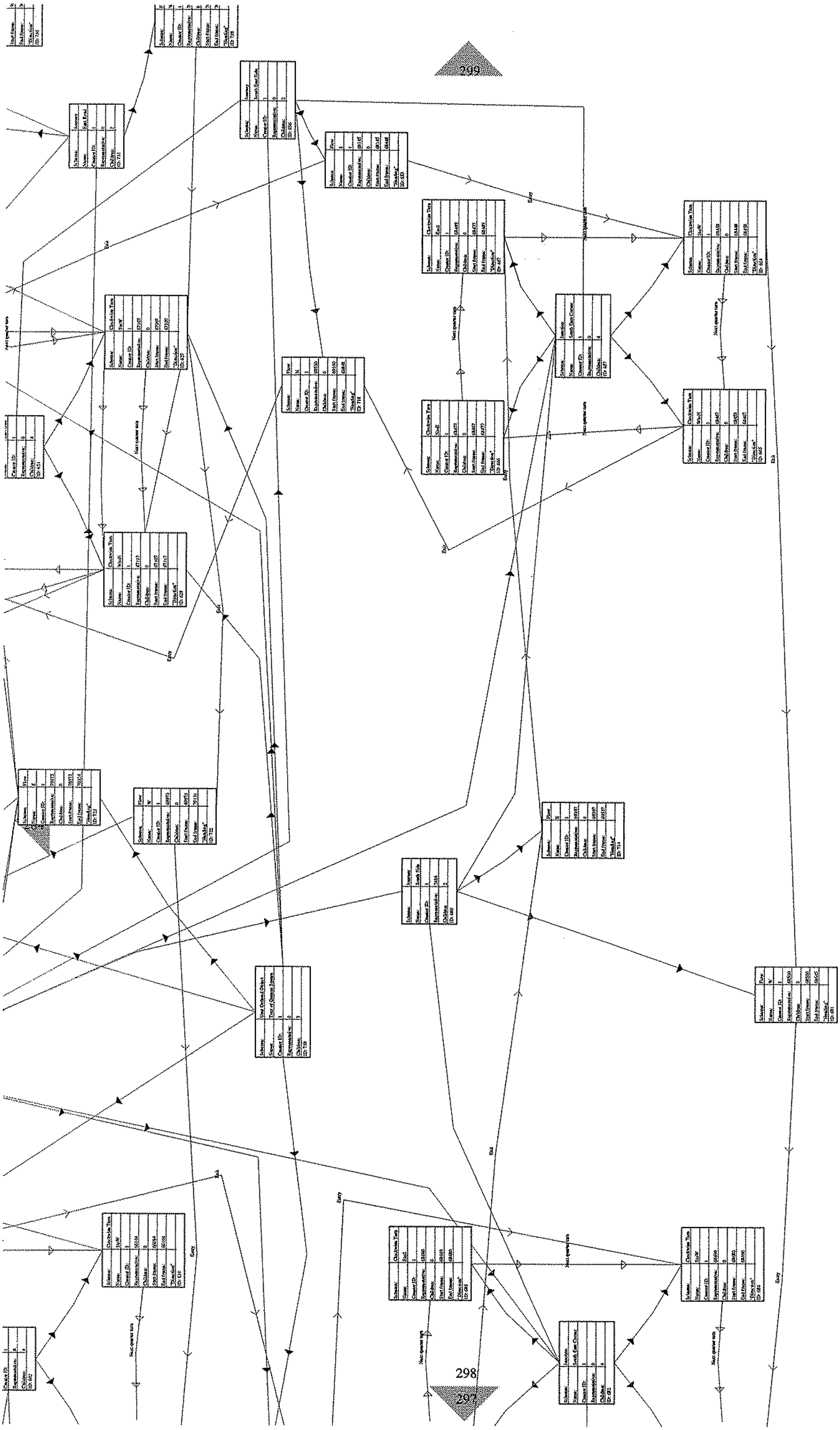

282
Schema: Scene
Name: Musical Scene
Creator ID: 1
Representative: 63331
Children: 0
Start frame: 63330
End frame: 63603
"Identifier" 2.m
"Approx date"
ID: 476
Schema: Scene
Name: Controversy about Citizen Kane
Creator ID: 1
Representative: 63640
Children: 0
Start frame: 63603
End frame: 64582
"Identifier" 2.n
"Approx date"
ID: 477
Schema: Part
Name: Welles about film
Creator ID: 1
Representative: 64585
Children: 0
Start frame: 64582
End frame: 64707
"Identifier" 3
ID: 478
Schema: Part
Name: Praise of Welles
Creator ID: 1
Representative: 64725
Children: 0
Start frame: 64707
End frame: 65393
"Identifier" 4
ID: 479
Schema: Part
Name: E
Creator ID: 1
Representative: 65550
Children: 0
Start frame: 65393
End frame: 65581
"Identifier" E
ID: 420
295
296
297

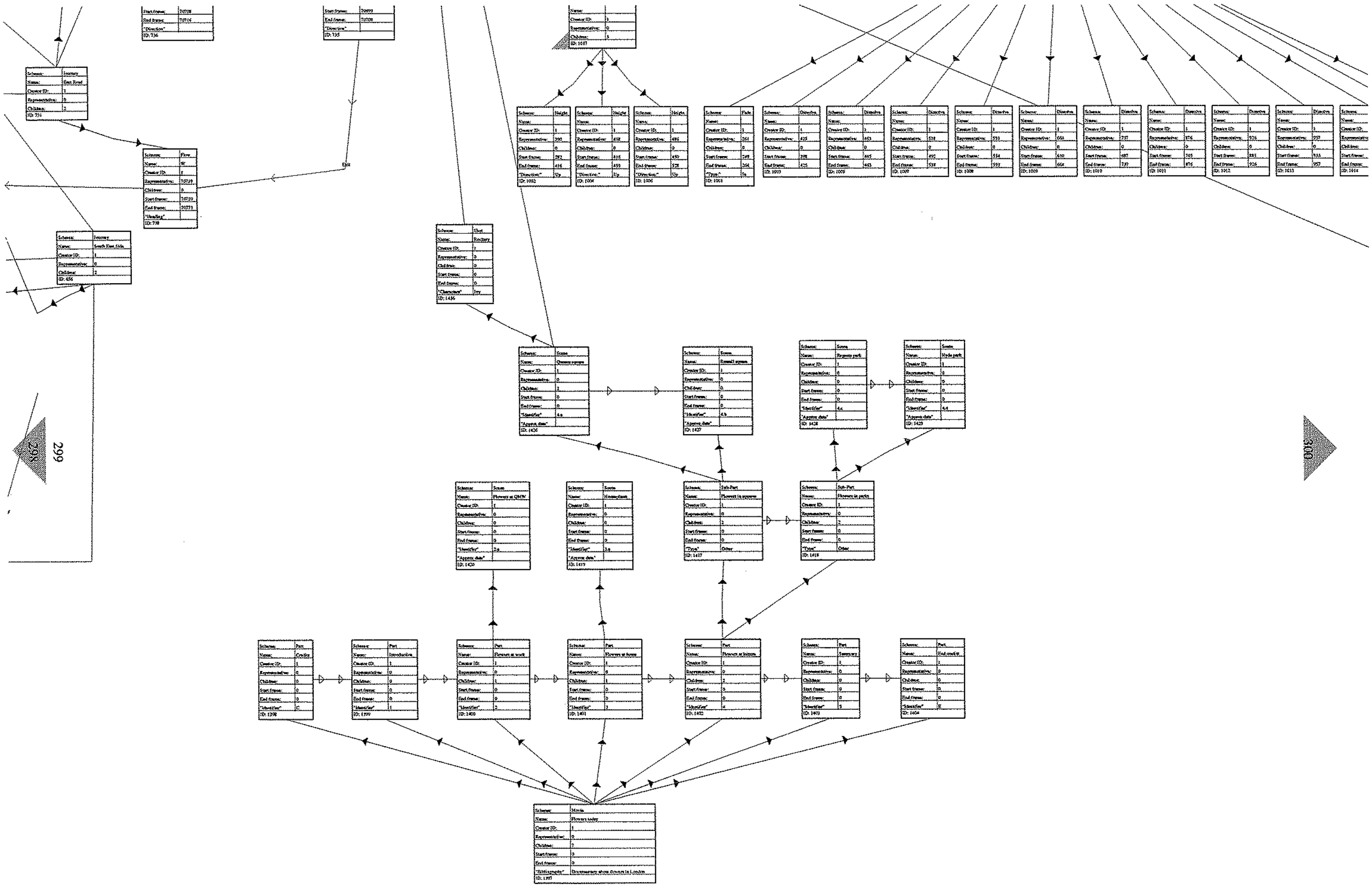
298

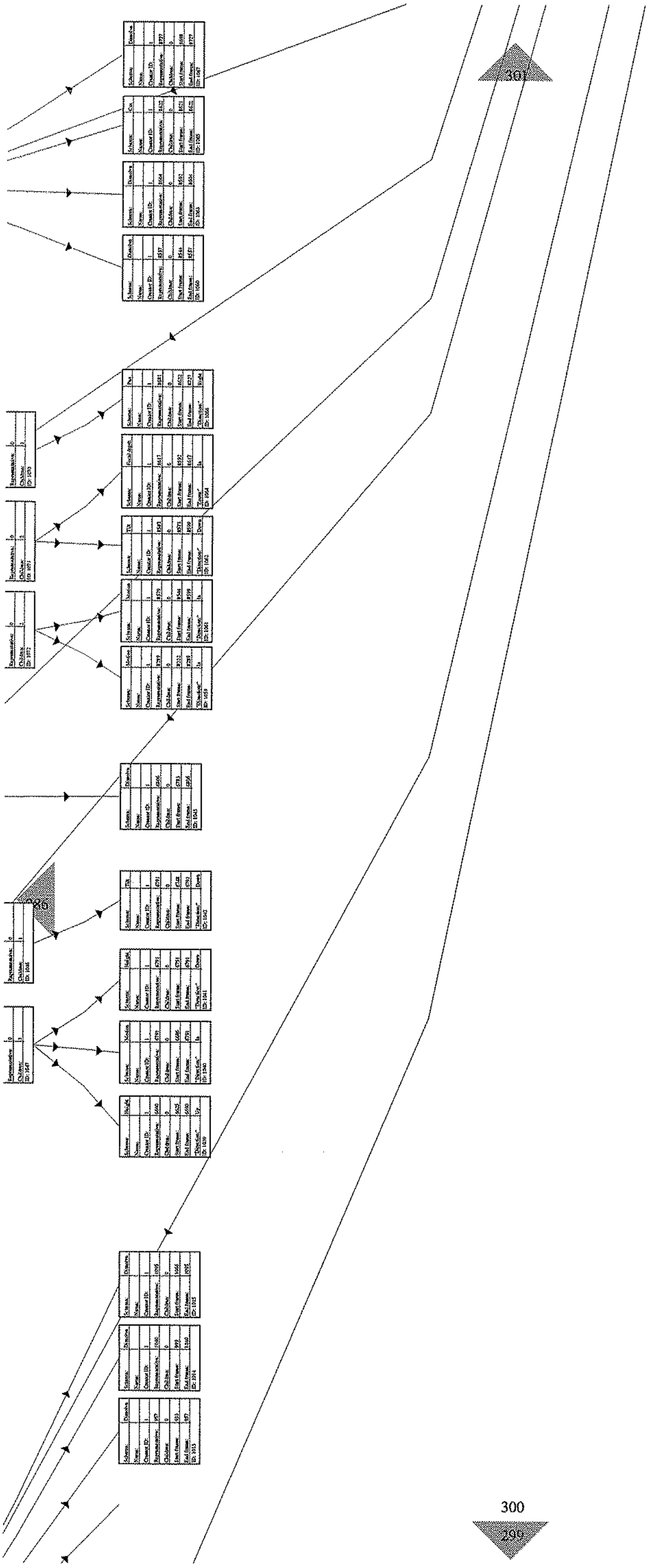

Schema: Clockwise Turn Name: SoW Creator ID: 1 Representative: 6610 Children: 0 Start frame: 6603 End frame: 6610 "Direction" ID: 613
Creator ID: 11 Representative: 0 Children: 4 ID: 602
Schema: Clockwise Turn Name: WtoN Creator ID: 1 Representative: 65115 Children: 0 Start frame: 65104 End frame: 65116 "Direction" ID: 623
Schema: Flow Name: K Creator ID: 1 Representative: 65738 Children: 0 Start frame: 65738 End frame: 65745 "Heading" ID: 703
Schema: Flow Name: S Creator ID: 1 Representative: 65066 Children: 0 Start frame: 65064 End frame: 65379 "Heading" ID: 704
Schema: Clockwise Turn Name: NtoE Creator ID: 1 Representative: 68659 Children: 0 Start frame: 68659 End frame: 68669 "Direction" ID: 684
Schema: Clockwise Name: EtoS Creator ID: 1 Representative: 68590 Children: 0 Start frame: 68589 End frame: 68590 "Direction" ID: 685
Schema: Junction Name: South East Corner Creator ID: 1 Representative: 0 Children: 4 ID: 682
Schema: Clockwise Name: StoW Creator ID: 1 Representative: 68690 Children: 0 Start frame: 68680 End frame: 68690 "Direction" ID: 686
Schema: Clockwise Turn Name: WtoS Creator ID: 1 Representative: 68659 Children: 0 Start frame: 68645 End frame: 68659 "Direction" ID: 683
Schema: Journey Name: South East Side Creator ID: 1 Representative: 0 Children: 2 ID: 702
Next quarter turn
Next quarter turn
Next quarter turn
Next quarter turn
Next quarter turn
Entry
Entry
Entry
Exit

283

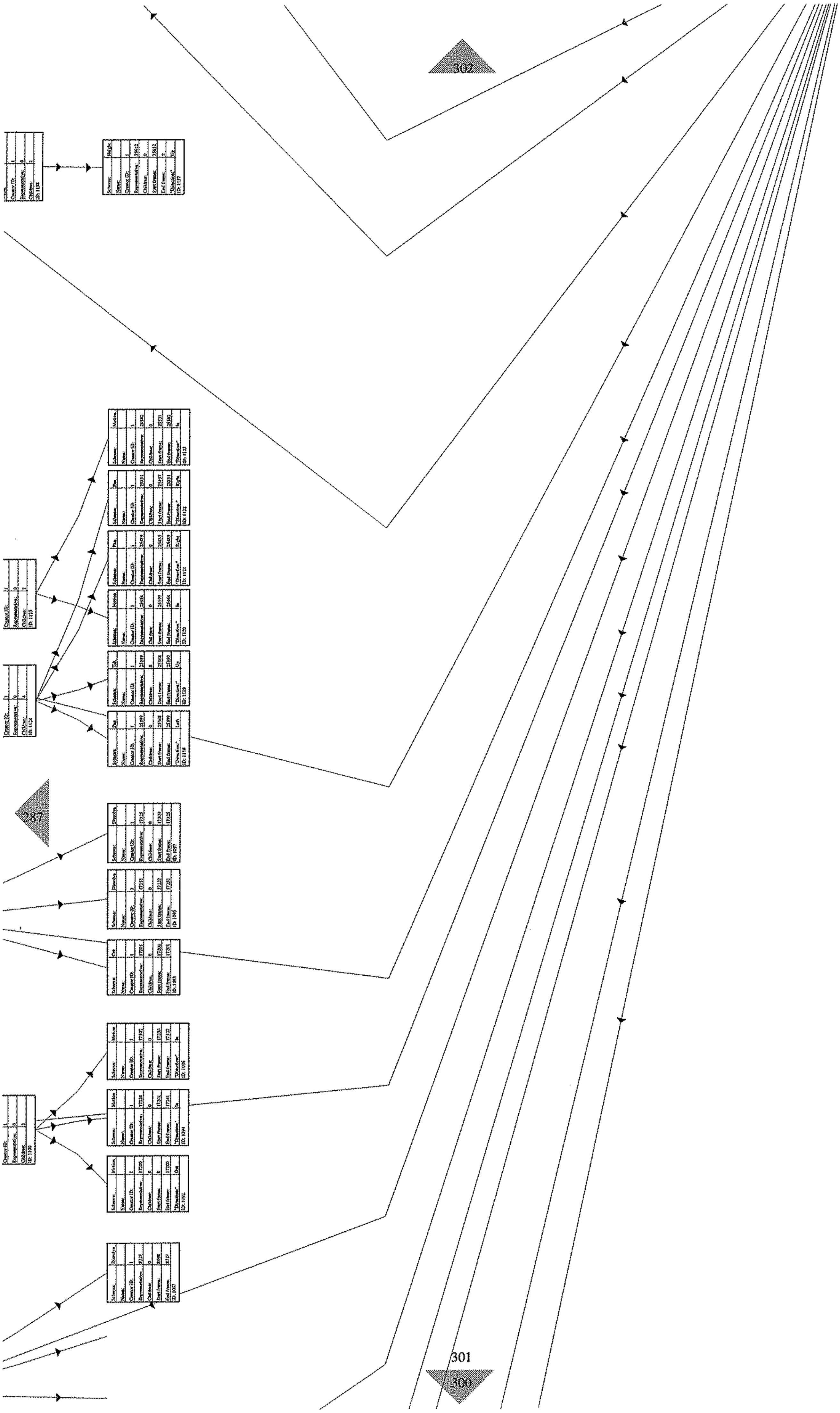
296

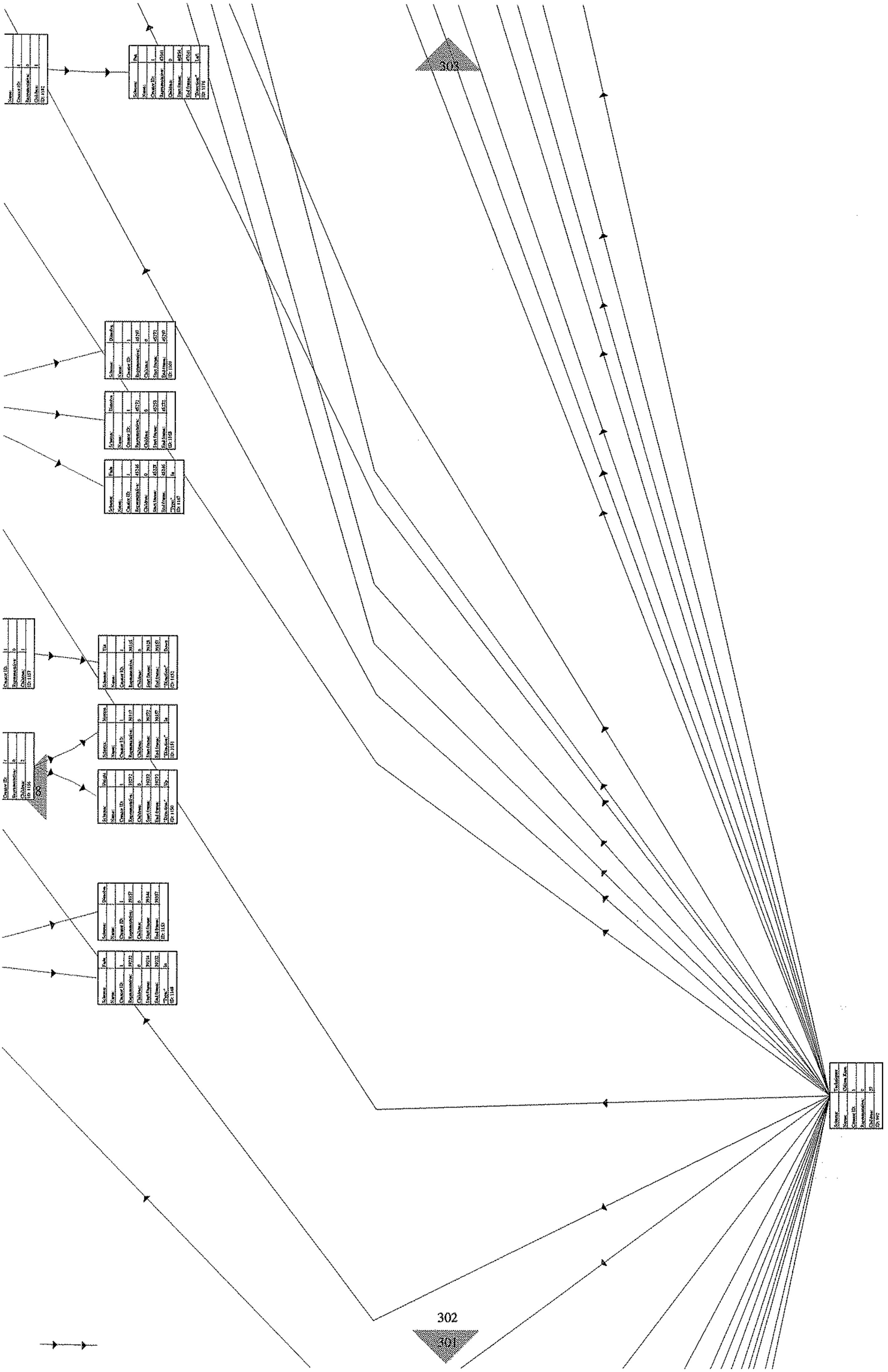
299
Next quarter turn
Entry
Exit
Schema: Junction
Name: South East Corner
ID: 657
Schema: Clockwise Turn
"Direction"
ID: 664
ID: 663
Schema: Journey
Name: South East Side
Schema: Flow
"Heading"
Schema: Journey
Name: South Side
Schema: User Defined Object
Name: Test of Quarter Square
298
297

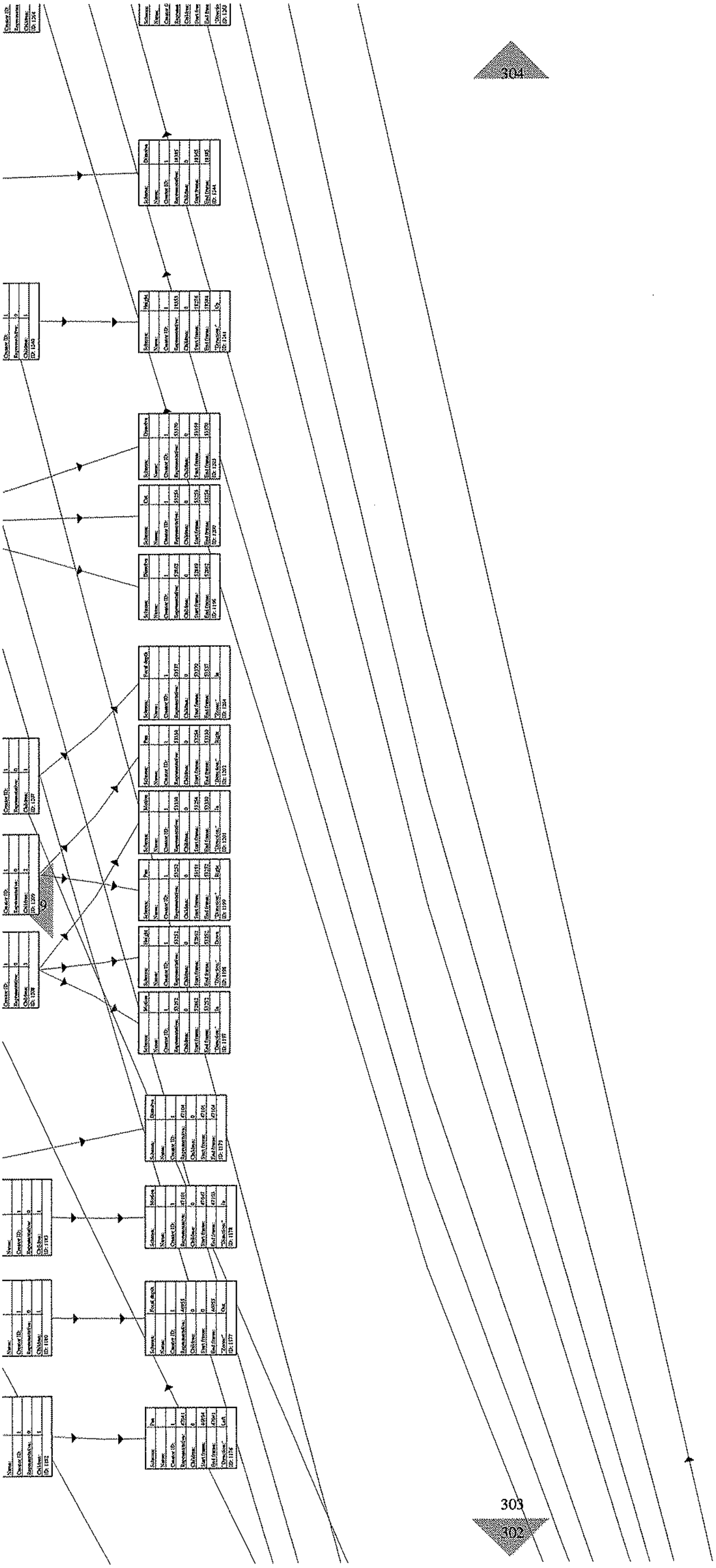

Schema: Movie | Name: Flowers today | Creator ID: 1 | Representative: 0 | Children: 7 | Start frame: 0 | End frame: 0 | "Bibliography" Documentary about flowers in London | ID: 1397
Schema: Part | Name: Credits | Creator ID: 1 | Representative: 0 | Children: 0 | Start frame: 0 | End frame: 0 | "Identifier" C | ID: 1398
Schema: Part | Name: Introduction | Creator ID: 1 | Representative: 0 | Children: 0 | Start frame: 0 | End frame: 0 | "Identifier" 1 | ID: 1399
Schema: Part | Name: Flowers at work | Creator ID: 1 | Representative: 0 | Children: 1 | Start frame: 0 | End frame: 0 | "Identifier" 2 | ID: 1400
Schema: Part | Name: Flowers at home | Creator ID: 1 | Representative: 0 | Children: 1 | Start frame: 0 | End frame: 0 | "Identifier" 3 | ID: 1401
Schema: Part | Name: Flowers at leisure | Creator ID: 1 | Representative: 0 | Children: 2 | Start frame: 0 | End frame: 0 | "Identifier" 4 | ID: 1402
Schema: Part | Name: Summary | Creator ID: 1 | Representative: 0 | Children: 0 | Start frame: 0 | End frame: 0 | "Identifier" 5 | ID: 1403
Schema: Part | Name: End credits | Creator ID: 1 | Representative: 0 | Children: 0 | Start frame: 0 | End frame: 0 | "Identifier" E | ID: 1404
Schema: Scene | Name: Flowers at QMW | Creator ID: 1 | Representative: 0 | Children: 0 | Start frame: 0 | End frame: 0 | "Identifier" 2.a | "Approx date" | ID: 1420
Schema: Scene | Name: Hampstead | Creator ID: 1 | Representative: 0 | Children: 0 | Start frame: 0 | End frame: 0 | "Identifier" 3.a | "Approx date" | ID: 1419
Schema: Sub-Part | Name: Flowers in squares | Creator ID: 1 | Representative: 0 | Children: 2 | Start frame: 0 | End frame: 0 | "Type" Other | ID: 1417
Schema: Sub-Part | Name: Flowers in parks | Creator ID: 1 | Representative: 0 | Children: 2 | Start frame: 0 | End frame: 0 | "Type" Other | ID: 1418
Schema: Scene | Name: Queens square | Creator ID: 1 | Representative: 0 | Children: 1 | Start frame: 0 | End frame: 0 | "Identifier" 4.a | "Approx date" | ID: 1426
Schema: Scene | Name: Russell square | Creator ID: 1 | Representative: 0 | Children: 0 | Start frame: 0 | End frame: 0 | "Identifier" 4.b | "Approx date" | ID: 1427
Schema: Scene | Name: Regents park | Creator ID: 1 | Representative: 0 | Children: 0 | Start frame: 0 | End frame: 0 | "Identifier" 4.c | "Approx date" | ID: 1428
Schema: Scene | Name: Hyde park | Creator ID: 1 | Representative: 0 | Children: 0 | Start frame: 0 | End frame: 0 | "Identifier" 4.d | "Approx date" | ID: 1429
Schema: Shot | Name: Rectory | Creator ID: 1 | Representative: 0 | Children: 0 | Start frame: 0 | End frame: 0 | "Character" Joy | ID: 1436
Schema: Journey | Name: East Road | Creator ID: 1 | Representative: 0 | Children: 2 | ID: 731
Schema: Flow | Name: W | Creator ID: 1 | Representative: 70739 | Children: 0 | Start frame: 70739 | End frame: 70773 | "HeadSeg" | ID: 738
Schema: Journey | Name: South East Side | Creator ID: 1 | Representative: 0 | Children: 2 | ID: 656
Start frame: 70708 | End frame: 70714 | "Direction" | ID: 736
Start frame: 70693 | End frame: 70708 | "Direction" | ID: 735
Exit
Name: | Creator ID: 1 | Representative: 0 | Children: 3 | ID: 1017
Schema: Height | Name: | Creator ID: 1 | Representative: 390 | Children: 0 | Start frame: 383 | End frame: 416 | "Direction" Up | ID: 1002
Schema: Height | Name: | Creator ID: 1 | Representative: 452 | Children: 0 | Start frame: 435 | End frame: 459 | "Direction" Up | ID: 1004
Schema: Height | Name: | Creator ID: 1 | Representative: 486 | Children: 0 | Start frame: 459 | End frame: 528 | "Direction" Up | ID: 1006
Schema: Fade | Name: | Creator ID: 1 | Representative: 264 | Children: 0 | Start frame: 249 | End frame: 264 | "Type" In | ID: 1001
Schema: Dissolve | Name: | Creator ID: 1 | Representative: 425 | Children: 0 | Start frame: 398 | End frame: 425 | ID: 1003
Schema: Dissolve | Name: | Creator ID: 1 | Representative: 463 | Children: 0 | Start frame: 445 | End frame: 463 | ID: 1005
Schema: Dissolve | Name: | Creator ID: 1 | Representative: 538 | Children: 0 | Start frame: 492 | End frame: 538 | ID: 1007
Schema: Dissolve | Name: | Creator ID: 1 | Representative: 593 | Children: 0 | Start frame: 554 | End frame: 593 | ID: 1008
Schema: Dissolve | Name: | Creator ID: 1 | Representative: 664 | Children: 0 | Start frame: 610 | End frame: 664 | ID: 1009
Schema: Dissolve | Name: | Creator ID: 1 | Representative: 737 | Children: 0 | Start frame: 687 | End frame: 737 | ID: 1010
Schema: Dissolve | Name: | Creator ID: 1 | Representative: 876 | Children: 0 | Start frame: 745 | End frame: 876 | ID: 1011
Schema: Dissolve | Name: | Creator ID: 1 | Representative: 926 | Children: 0 | Start frame: 883 | End frame: 926 | ID: 1012
Schema: Dissolve | Name: | Creator ID: 1 | Representative: 957 | Children: 0 | Start frame: 933 | End frame: 957 | ID: 1013
Schema: | Name: | Creator ID: | Representative: | Children: | Start frame: | End frame: | ID: 1014
298
300

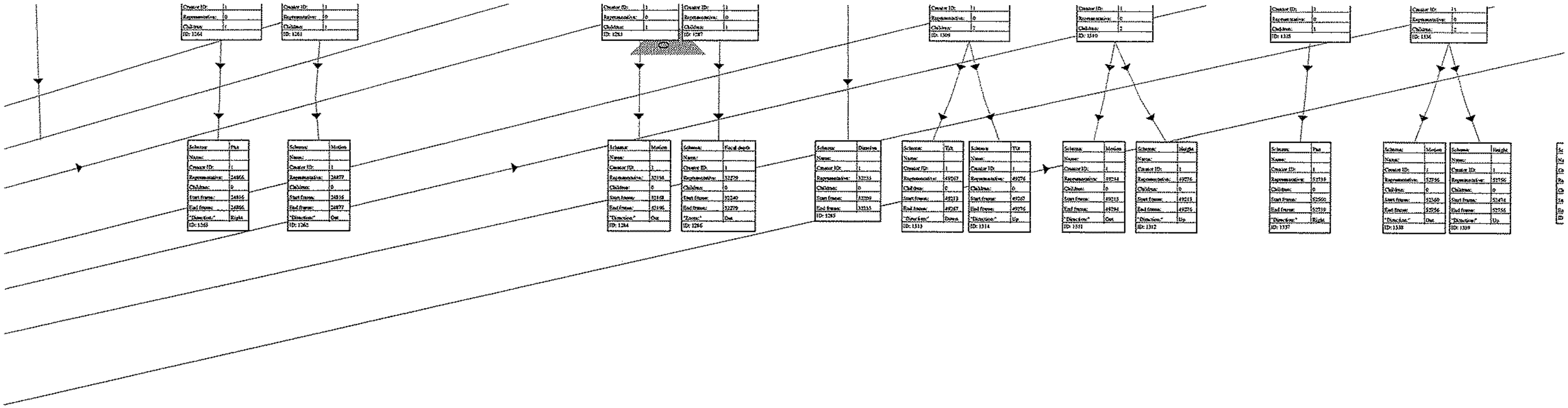


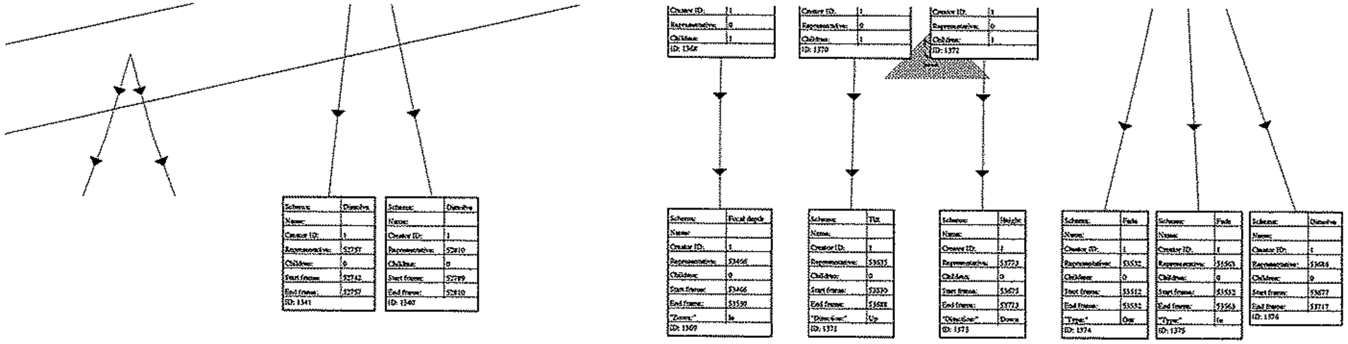
301
298
300
299


302
287
301
300


303
302
301

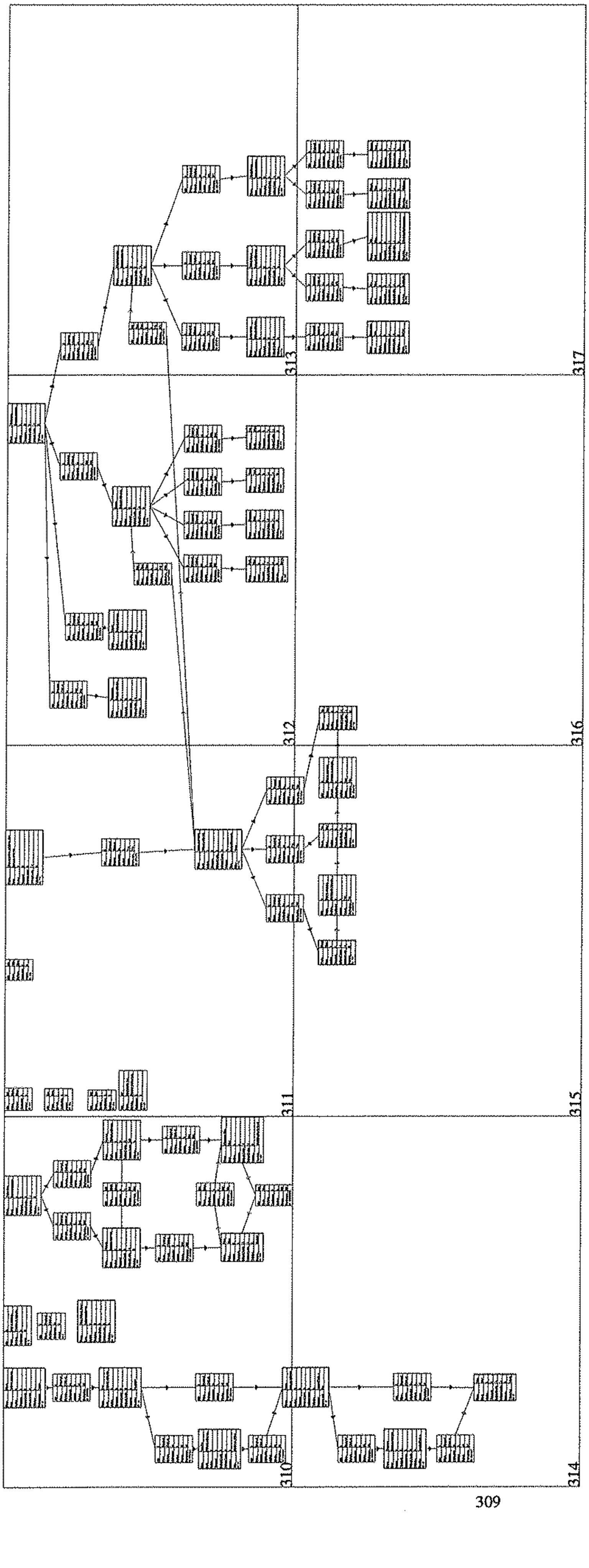

304
303
302


Creator ID: 1
Representative: 0
Children: 1
ID: 1264
Creator ID: 1
Representative: 0
Children: 1
ID: 1261
Schema: Pan
Name:
Creator ID: 1
Representative: 24866
Children: 0
Start frame: 24816
End frame: 24866
"Direction:" Right
ID: 1263
Schema: Motion
Name:
Creator ID: 1
Representative: 24877
Children: 0
Start frame: 24816
End frame: 24877
"Direction:" Out
ID: 1262
Creator ID: 1
Representative: 0
Children: 1
ID: 1283
Creator ID: 1
Representative: 0
Children: 1
ID: 1287
Schema: Motion
Name:
Creator ID: 1
Representative: 32158
Children: 0
Start frame: 32158
End frame: 45196
"Direction:" Out
ID: 1284
Schema: Focal depth
Name:
Creator ID: 1
Representative: 32279
Children: 0
Start frame: 32240
End frame: 32279
"Zoom:" Out
ID: 1286
Schema: Dissolve
Name:
Creator ID: 1
Representative: 32233
Children: 0
Start frame: 32209
End frame: 32233
ID: 1285
Creator ID: 1
Representative: 0
Children: 2
ID: 1309
Creator ID: 1
Representative: 0
Children: 2
ID: 1310
Schema: Tilt
Name:
Creator ID: 1
Representative: 49267
Children: 0
Start frame: 49213
End frame: 49267
"Direction:" Down
ID: 1313
Schema: Tilt
Name:
Creator ID: 1
Representative: 49276
Children: 0
Start frame: 49267
End frame: 49276
"Direction:" Up
ID: 1314
Schema: Motion
Name:
Creator ID: 1
Representative: 49294
Children: 0
Start frame: 49213
End frame: 49294
"Direction:" Out
ID: 1311
Schema: Height
Name:
Creator ID: 1
Representative: 49276
Children: 0
Start frame: 49213
End frame: 49276
"Direction:" Up
ID: 1312
Creator ID: 1
Representative: 0
Children: 1
ID: 1335
Creator ID: 1
Representative: 0
Children: 2
ID: 1336
Schema: Pan
Name:
Creator ID: 1
Representative: 52739
Children: 0
Start frame: 52560
End frame: 52739
"Direction:" Right
ID: 1337
Schema: Motion
Name:
Creator ID: 1
Representative: 52756
Children: 0
Start frame: 52560
End frame: 52756
"Direction:" Out
ID: 1338
Schema: Height
Name:
Creator ID: 1
Representative: 52756
Children: 0
Start frame: 52474
End frame: 52756
"Direction:" Up
ID: 1339

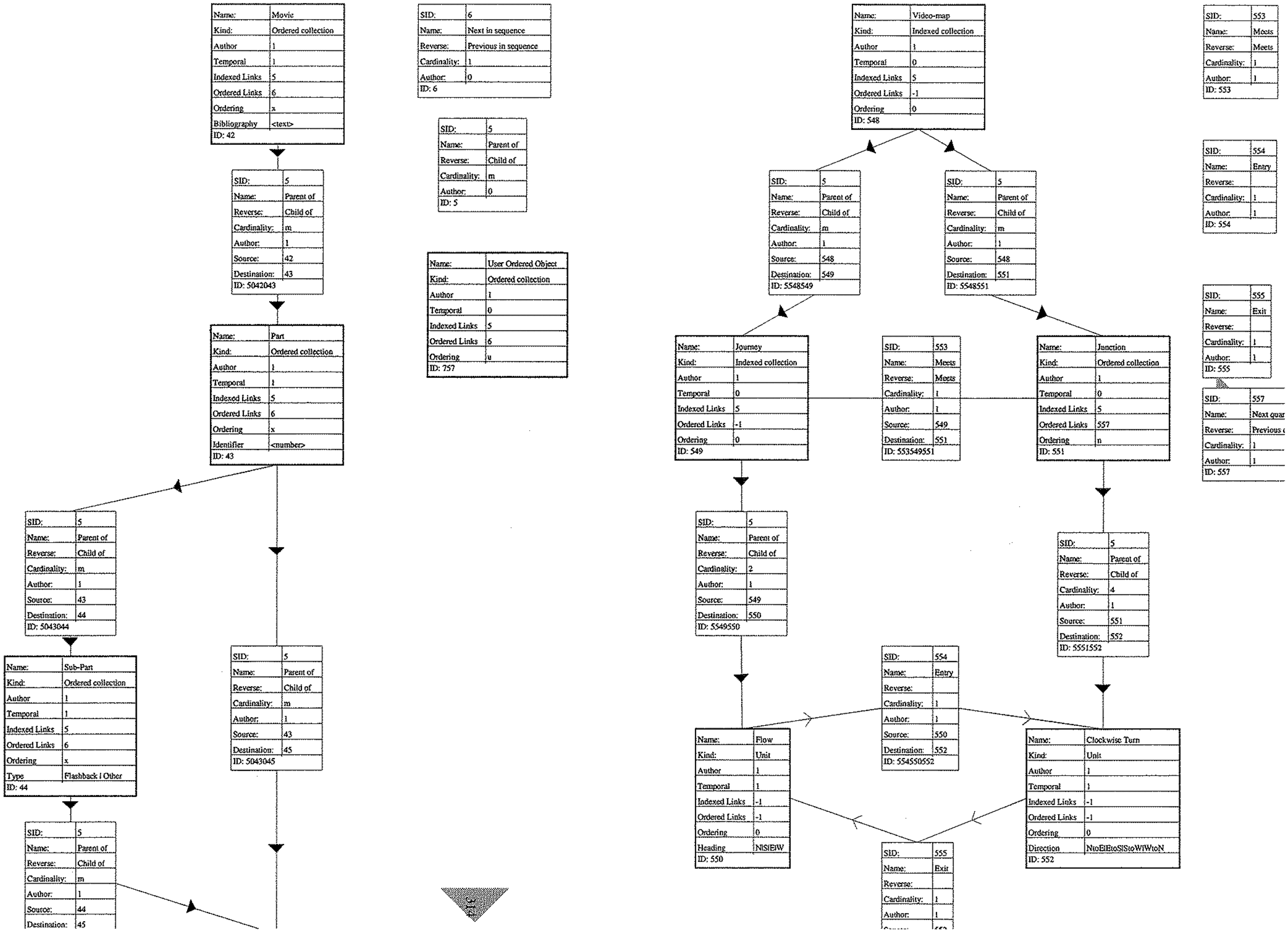
Creator ID: 1
Representative: 0
Children: 1
ID: 1368
Creator ID: 1
Representative: 0
Children: 1
ID: 1370
Creator ID: 1
Representative: 0
Children: 1
ID: 1372
Scheme: Dissolve
Name:
Creator ID: 1
Representative: 52757
Children: 0
Start frame: 52742
End frame: 52757
ID: 1341
Scheme: Dissolve
Name:
Creator ID: 1
Representative: 52810
Children: 0
Start frame: 52789
End frame: 52810
ID: 1340
Scheme: Focal depth
Name:
Creator ID: 1
Representative: 53466
Children: 0
Start frame: 53466
End frame: 53530
"Zoom:" In
ID: 1369
Scheme: Tilt
Name:
Creator ID: 1
Representative: 53535
Children: 0
Start frame: 53530
End frame: 53588
"Direction:" Up
ID: 1371
Scheme: Height
Name:
Creator ID: 1
Representative: 53773
Children: 0
Start frame: 53675
End frame: 53773
"Direction:" Down
ID: 1373
Scheme: Fade
Name:
Creator ID: 1
Representative: 53532
Children: 0
Start frame: 53512
End frame: 53532
"Type:" Out
ID: 1374
Scheme: Fade
Name:
Creator ID: 1
Representative: 53563
Children: 0
Start frame: 53532
End frame: 53563
"Type:" In
ID: 1375
Scheme: Dissolve
Name:
Creator ID: 1
Representative: 53686
Children: 0
Start frame: 53677
End frame: 53717
ID: 1376

Appendix E VCMF Schemas

This appendix gives a graphical representation of the schemas used in the demonstration. The next page contains an overview of the rest of the appendix; the diagram is divided into squares which represent each page (indicated by a page number). On further pages grey triangles indicate appropriate adjacent pages.



310
311
312
313
314
315
316
317



Name: Movie
Kind: Ordered collection
Author 1
Temporal 1
Indexed Links 5
Ordered Links 6
Ordering x
Bibliography <text>
ID: 42

SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 42
Destination: 43
ID: 5042043

Name: Part
Kind: Ordered collection
Author 1
Temporal 1
Indexed Links 5
Ordered Links 6
Ordering x
Identifier <number>
ID: 43

SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 43
Destination: 44
ID: 5043044

Name: Sub-Part
Kind: Ordered collection
Author 1
Temporal 1
Indexed Links 5
Ordered Links 6
Ordering x
Type Flashback | Other
ID: 44

SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 44
Destination: 45

SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 43
Destination: 45
ID: 5043045

SID: 6
Name: Next in sequence
Reverse: Previous in sequence
Cardinality: 1
Author: 0
ID: 6

SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 0
ID: 5

Name: User Ordered Object
Kind: Ordered collection
Author 1
Temporal 0
Indexed Links 5
Ordered Links 6
Ordering u
ID: 757

Name: Video-map
Kind: Indexed collection
Author 1
Temporal 0
Indexed Links 5
Ordered Links -1
Ordering 0
ID: 548

SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 548
Destination: 549
ID: 5548549

SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 548
Destination: 551
ID: 5548551

Name: Journey
Kind: Indexed collection
Author 1
Temporal 0
Indexed Links 5
Ordered Links -1
Ordering 0
ID: 549

SID: 553
Name: Meets
Reverse: Meets
Cardinality: 1
Author: 1
Source: 549
Destination: 551
ID: 553549551

Name: Junction
Kind: Ordered collection
Author 1
Temporal 0
Indexed Links 5
Ordered Links 557
Ordering n
ID: 551

SID: 5
Name: Parent of
Reverse: Child of
Cardinality: 2
Author: 1
Source: 549
Destination: 550
ID: 5549550

SID: 5
Name: Parent of
Reverse: Child of
Cardinality: 4
Author: 1
Source: 551
Destination: 552
ID: 5551552

SID: 554
Name: Entry
Reverse:
Cardinality: 1
Author: 1
Source: 550
Destination: 552
ID: 554550552

Name: Flow
Kind: Unit
Author 1
Temporal 1
Indexed Links -1
Ordered Links -1
Ordering 0
Heading N|S|E|W
ID: 550

Name: Clockwise Turn
Kind: Unit
Author 1
Temporal 1
Indexed Links -1
Ordered Links -1
Ordering 0
Direction NtoE|EtoS|StoW|WtoN
ID: 552

SID: 555
Name: Exit
Reverse:
Cardinality: 1
Author: 1

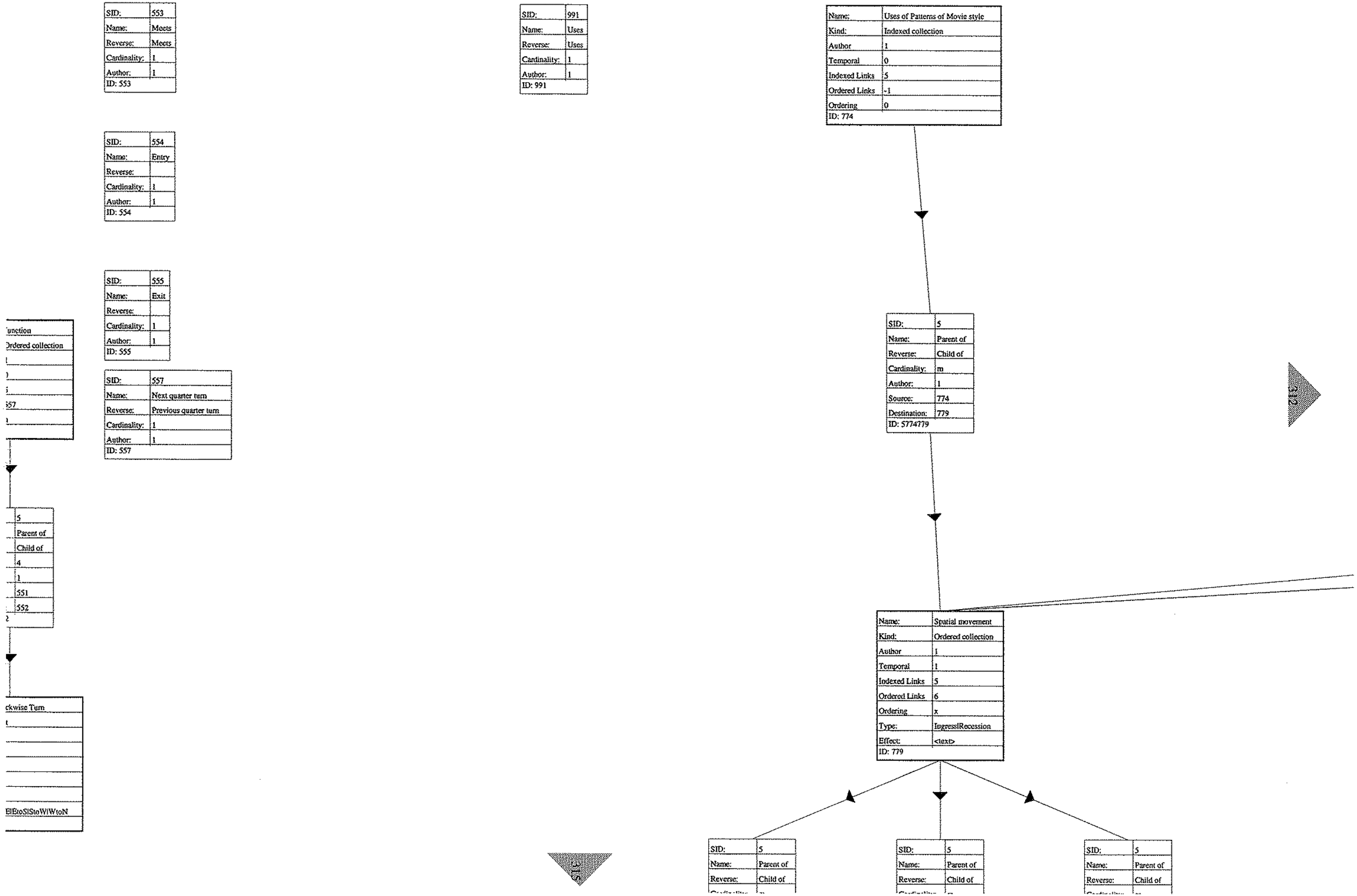
SID: 553
Name: Meets
Reverse: Meets
Cardinality: 1
Author: 1
ID: 553

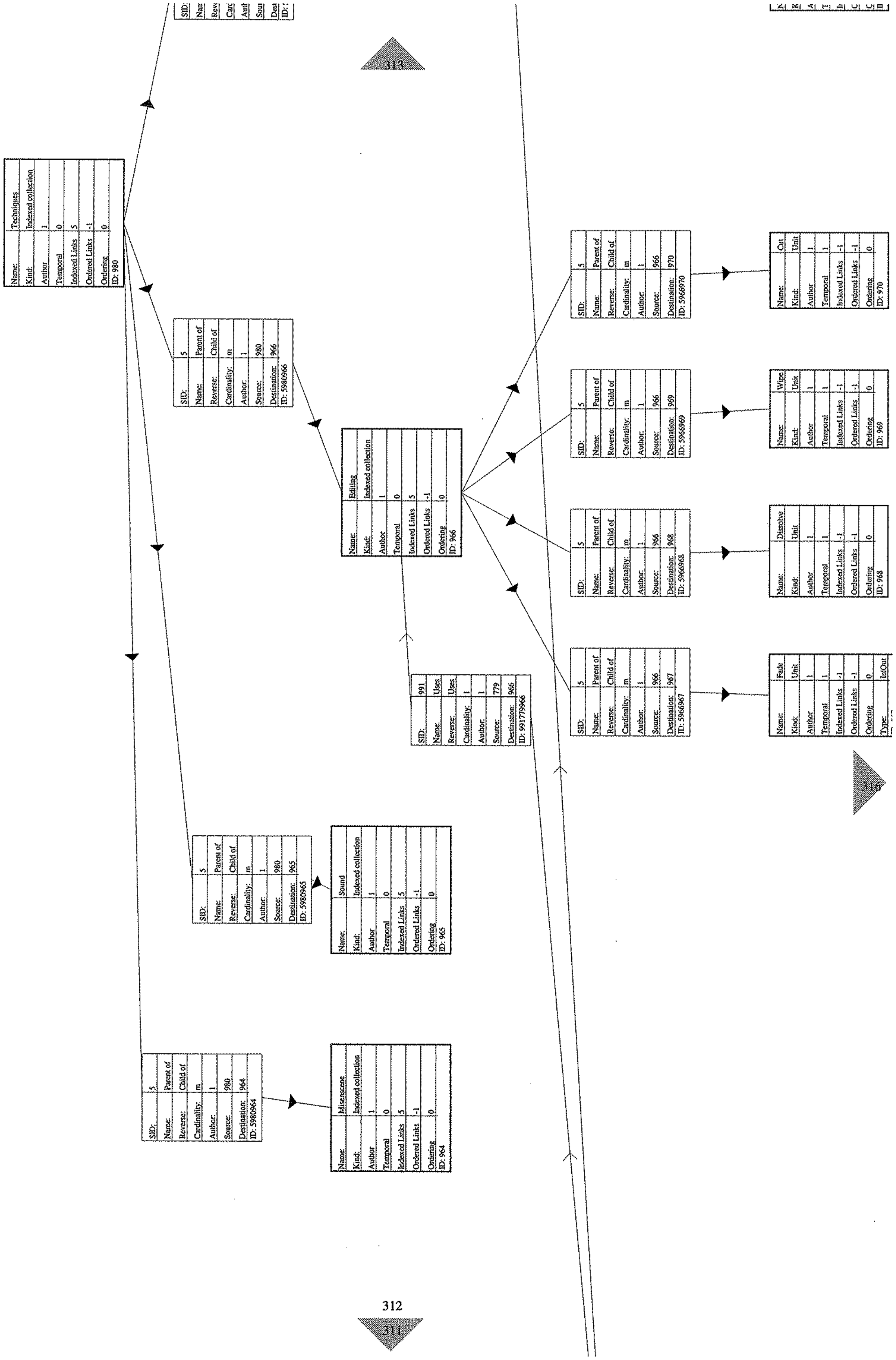
SID: 554
Name: Entry
Reverse:
Cardinality: 1
Author: 1
ID: 554

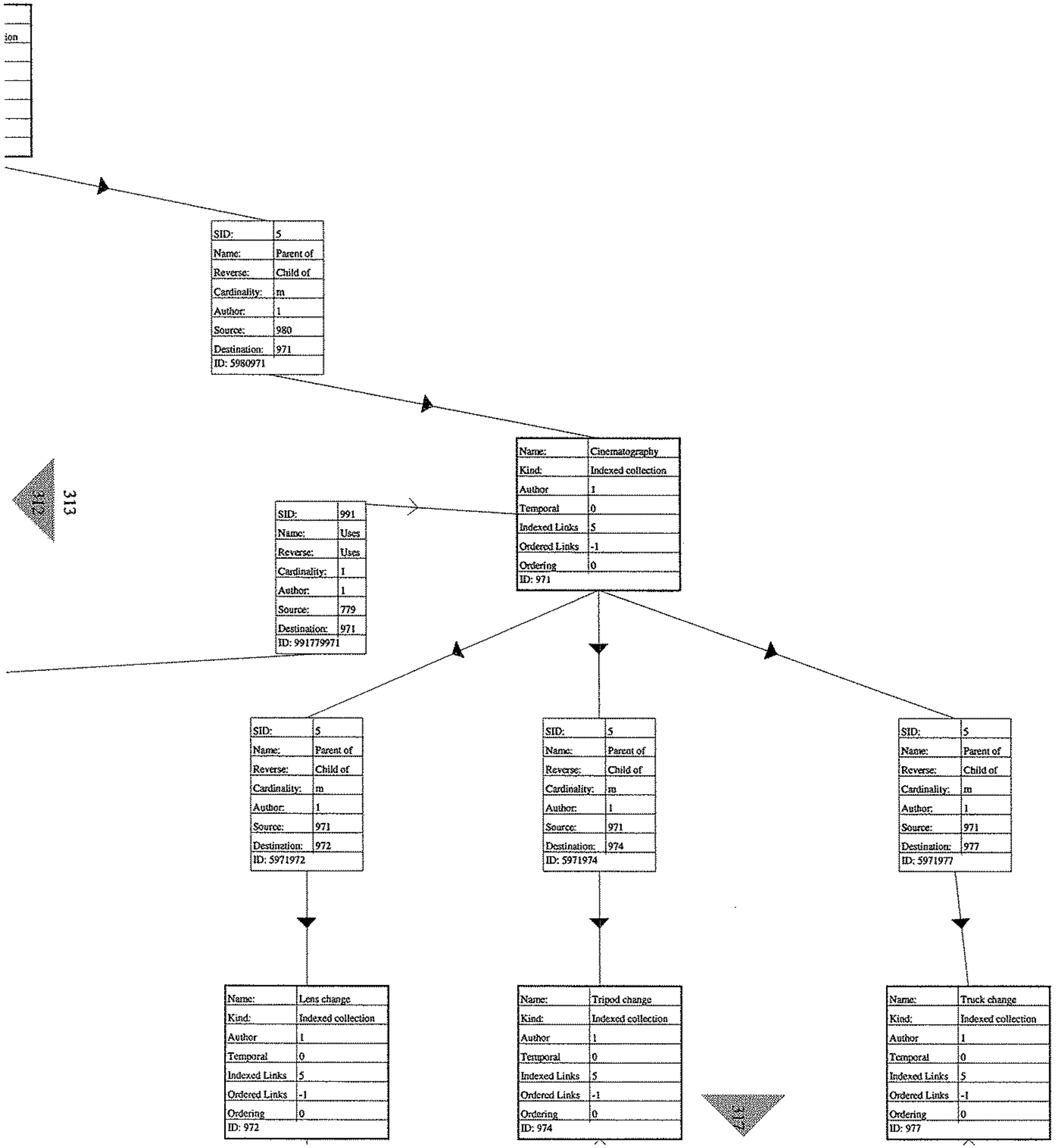
SID: 555
Name: Exit
Reverse:
Cardinality: 1
Author: 1
ID: 555

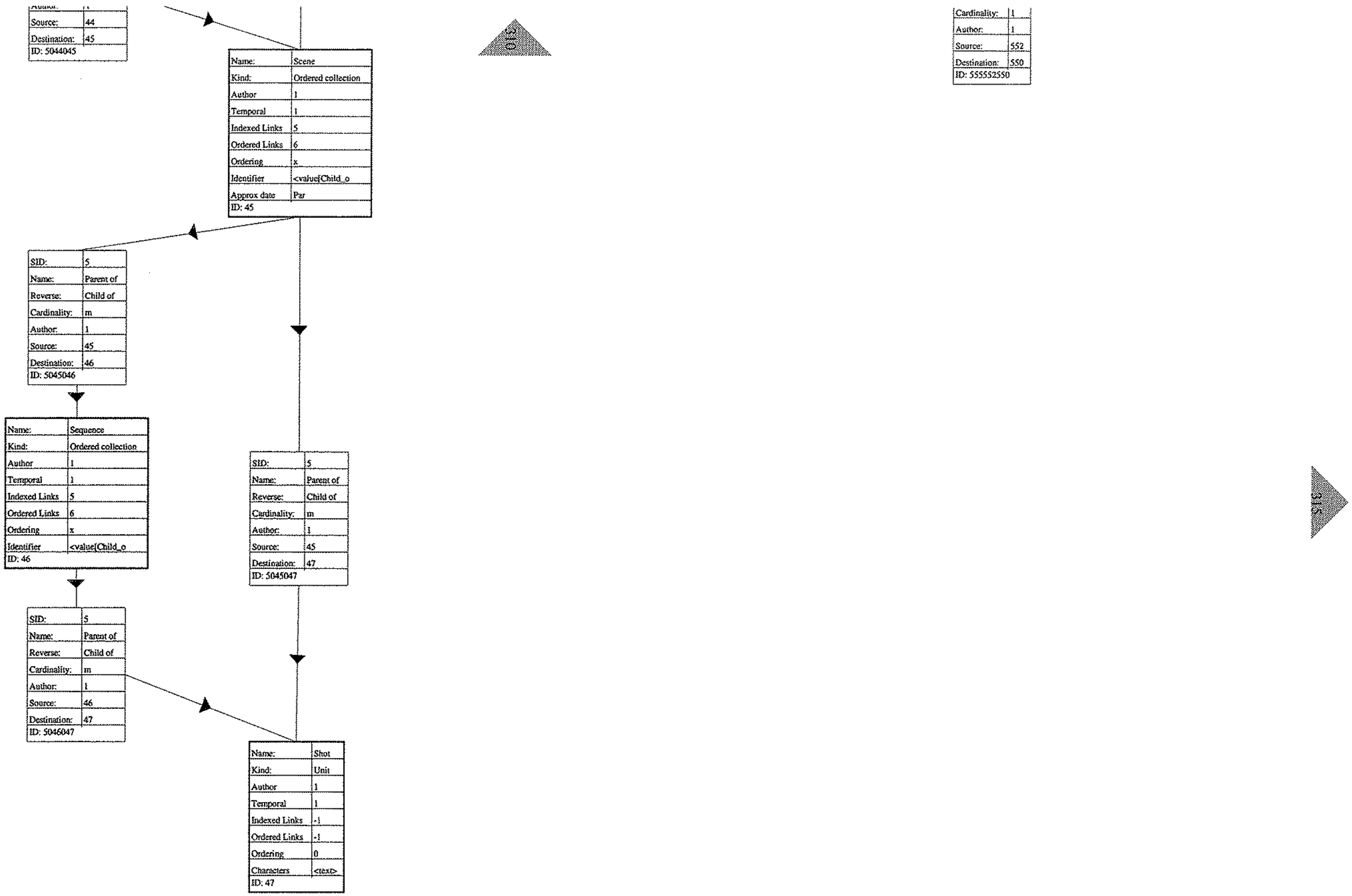
SID: 557
Name: Next quar
Reverse: Previous (
Cardinality: 1
Author: 1
ID: 557

314


SID: 553
Name: Meets
Reverse: Meets
Cardinality: 1
Author: 1
ID: 553
SID: 554
Name: Entry
Reverse:
Cardinality: 1
Author: 1
ID: 554
SID: 555
Name: Exit
Reverse:
Cardinality: 1
Author: 1
ID: 555
SID: 557
Name: Next quarter turn
Reverse: Previous quarter turn
Cardinality: 1
Author: 1
ID: 557
SID: 991
Name: Uses
Reverse: Uses
Cardinality: 1
Author: 1
ID: 991
Name: Uses of Patterns of Movie style
Kind: Indexed collection
Author 1
Temporal 0
Indexed Links 5
Ordered Links -1
Ordering 0
ID: 774
SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 774
Destination: 779
ID: 5774779
312
Name: Spatial movement
Kind: Ordered collection
Author 1
Temporal 1
Indexed Links 5
Ordered Links 6
Ordering x
Type: Ingress|Recession
Effect: <text>
ID: 779
SID: 5
Name: Parent of
Reverse: Child of
SID: 5
Name: Parent of
Reverse: Child of
SID: 5
Name: Parent of
Reverse: Child of
315
'unction
Ordered collection
557
5
Parent of
Child of
4
1
551
552
2
ckwise Turn
E|EtoS|StoW|WtoN


Name: Techniques
Kind: Indexed collection
Author 1
Temporal 0
Indexed Links 5
Ordered Links -1
Ordering 0
ID: 980
SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 980
Destination: 966
ID: 5980966
Name: Editing
Kind: Indexed collection
Author 1
Temporal 0
Indexed Links 5
Ordered Links -1
Ordering 0
ID: 966
SID: 991
Name: Uses
Reverse: Uses
Cardinality: 1
Author: 1
Source: 779
Destination: 966
ID: 991779966
SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 966
Destination: 970
ID: 5966970
Name: Cut
Kind: Unit
Author 1
Temporal 1
Indexed Links -1
Ordered Links -1
Ordering 0
ID: 970
SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 966
Destination: 969
ID: 5966969
Name: Wipe
Kind: Unit
Author 1
Temporal 1
Indexed Links -1
Ordered Links -1
Ordering 0
ID: 969
SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 966
Destination: 968
ID: 5966968
Name: Dissolve
Kind: Unit
Author 1
Temporal 1
Indexed Links -1
Ordered Links -1
Ordering 0
ID: 968
SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 966
Destination: 967
ID: 5966967
Name: Fade
Kind: Unit
Author 1
Temporal 1
Indexed Links -1
Ordered Links -1
Ordering 0
Type: InOut
SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 980
Destination: 965
ID: 5980965
Name: Sound
Kind: Indexed collection
Author 1
Temporal 0
Indexed Links 5
Ordered Links -1
Ordering 0
ID: 965
SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 980
Destination: 964
ID: 5980964
Name: Misenscene
Kind: Indexed collection
Author 1
Temporal 0
Indexed Links 5
Ordered Links -1
Ordering 0
ID: 964
313
316
312
311


ion
SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 980
Destination: 971
ID: 5980971
312
SID: 991
Name: Uses
Reverse: Uses
Cardinality: 1
Author: 1
Source: 779
Destination: 971
ID: 991779971
Name: Cinematography
Kind: Indexed collection
Author 1
Temporal 0
Indexed Links 5
Ordered Links -1
Ordering 0
ID: 971
SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 971
Destination: 972
ID: 5971972
SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 971
Destination: 974
ID: 5971974
SID: 5
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 971
Destination: 977
ID: 5971977
Name: Lens change
Kind: Indexed collection
Author 1
Temporal 0
Indexed Links 5
Ordered Links -1
Ordering 0
ID: 972
Name: Tripod change
Kind: Indexed collection
Author 1
Temporal 0
Indexed Links 5
Ordered Links -1
Ordering 0
ID: 974
317
Name: Truck change
Kind: Indexed collection
Author 1
Temporal 0
Indexed Links 5
Ordered Links -1
Ordering 0
ID: 977



Author:	1
Source:	44
Destination:	45
ID: 5044045	

Name:	Scene
Kind:	Ordered collection
Author	1
Temporal	1
Indexed Links	5
Ordered Links	6
Ordering	x
Identifier	<value[Child_o
Approx date	Par
ID: 45	

SID:	5
Name:	Parent of
Reverse:	Child of
Cardinality:	m
Author:	1
Source:	45
Destination:	46
ID: 5045046	

Name:	Sequence
Kind:	Ordered collection
Author	1
Temporal	1
Indexed Links	5
Ordered Links	6
Ordering	x
Identifier	<value[Child_o
ID: 46	

SID:	5
Name:	Parent of
Reverse:	Child of
Cardinality:	m
Author:	1
Source:	46
Destination:	47
ID: 5046047	

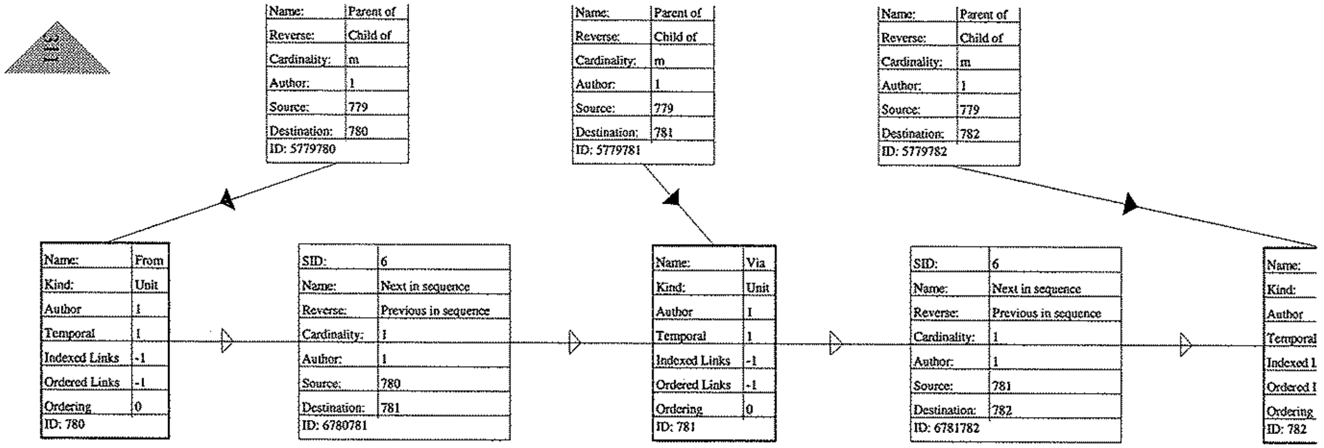
SID:	5
Name:	Parent of
Reverse:	Child of
Cardinality:	m
Author:	1
Source:	45
Destination:	47
ID: 5045047	

Name:	Shot
Kind:	Unit
Author	1
Temporal	1
Indexed Links	-1
Ordered Links	-1
Ordering	0
Characters	<text>
ID: 47	

Cardinality:	1
Author:	1
Source:	552
Destination:	550
ID: 555552550	

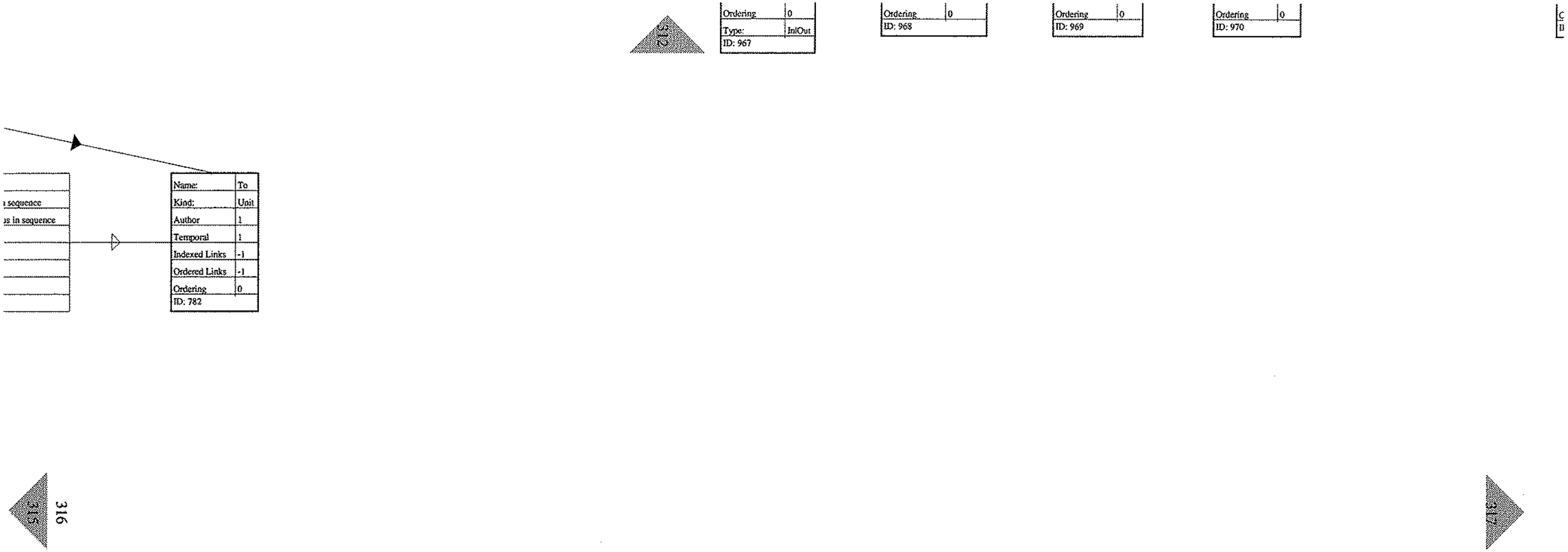
314

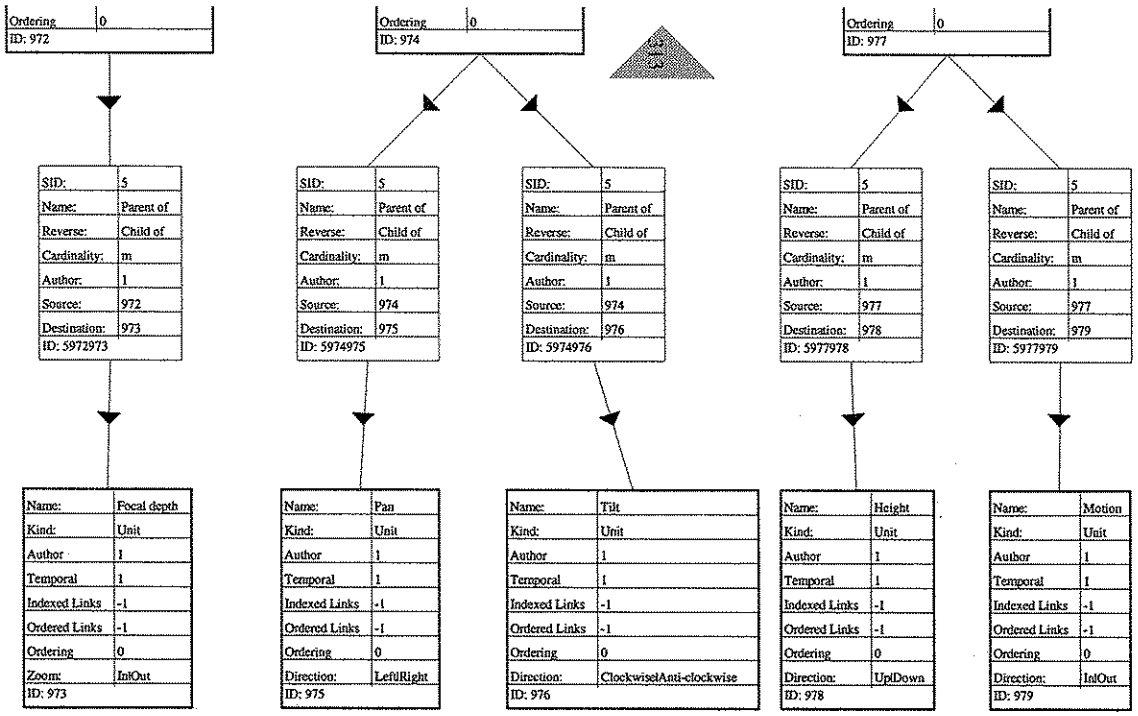
311


Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 779
Destination: 780
ID: 5779780
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 779
Destination: 781
ID: 5779781
Name: Parent of
Reverse: Child of
Cardinality: m
Author: 1
Source: 779
Destination: 782
ID: 5779782
Name: From
Kind: Unit
Author 1
Temporal 1
Indexed Links -1
Ordered Links -1
Ordering 0
ID: 780
SID: 6
Name: Next in sequence
Reverse: Previous in sequence
Cardinality: 1
Author: 1
Source: 780
Destination: 781
ID: 6780781
Name: Via
Kind: Unit
Author 1
Temporal 1
Indexed Links -1
Ordered Links -1
Ordering 0
ID: 781
SID: 6
Name: Next in sequence
Reverse: Previous in sequence
Cardinality: 1
Author: 1
Source: 781
Destination: 782
ID: 6781782
Name:
Kind:
Author
Temporal
Indexed L
Ordered I
Ordering
ID: 782

316

314


Ordering 0
Type: In/Out
ID: 967
Ordering 0
ID: 968
Ordering 0
ID: 969
Ordering 0
ID: 970
312
317
315
a sequence
ns in sequence
Name: To
Kind: Unit
Author 1
Temporal 1
Indexed Links -1
Ordered Links -1
Ordering 0
ID: 782



Ordering	0
ID: 972	

SID:	5
Name:	Parent of
Reverse:	Child of
Cardinality:	m
Author:	1
Source:	972
Destination:	973
ID: 5972973	

Name:	Focal depth
Kind:	Unit
Author	1
Temporal	1
Indexed Links	-1
Ordered Links	-1
Ordering	0
Zoom:	In|Out
ID: 973	

Ordering	0
ID: 974	

SID:	5
Name:	Parent of
Reverse:	Child of
Cardinality:	m
Author:	1
Source:	974
Destination:	975
ID: 5974975	

Name:	Pan
Kind:	Unit
Author	1
Temporal	1
Indexed Links	-1
Ordered Links	-1
Ordering	0
Direction:	Left|Right
ID: 975	

SID:	5
Name:	Parent of
Reverse:	Child of
Cardinality:	m
Author:	1
Source:	974
Destination:	976
ID: 5974976	

Name:	Tilt
Kind:	Unit
Author	1
Temporal	1
Indexed Links	-1
Ordered Links	-1
Ordering	0
Direction:	Clockwise|Anti-clockwise
ID: 976	

Ordering	0
ID: 977	

SID:	5
Name:	Parent of
Reverse:	Child of
Cardinality:	m
Author:	1
Source:	977
Destination:	978
ID: 5977978	

Name:	Height
Kind:	Unit
Author	1
Temporal	1
Indexed Links	-1
Ordered Links	-1
Ordering	0
Direction:	Up|Down
ID: 978	

SID:	5
Name:	Parent of
Reverse:	Child of
Cardinality:	m
Author:	1
Source:	977
Destination:	979
ID: 5977979	

Name:	Motion
Kind:	Unit
Author	1
Temporal	1
Indexed Links	-1
Ordered Links	-1
Ordering	0
Direction:	In|Out
ID: 979	