Quality of Service Specification in Video Databases

Elisa Bertino* **Tiziana Catarci†** **Ahmed K. Elmagarmid‡** **Mohand-Saïd Hacid§**

*Dipartimento di Scienze dell'Informazione
University of Milano
Via Comelico, 39/41 20135 Milani - Italy
bertino@dsi.unimi.it

†Dipartimento di Informatica e Sistemistica
Universita' degli Studi di Roma "La Sapienza"
Via Salaria 113, 00198 Roma - Italy
catarci@dis.uniroma1.it

‡Department of Computer Sciences
Purdue University
West Lafayette, IN 47907 - USA
ake@cs.purdue.edu

§LIRIS - UFR Informatique
Université Lyon 1
43, blvd du 11 novembre 1918
69622 Villeurbanne - France
mshacid@bat710.univ-lyon1.fr

Abstract

Quality of Service (QoS) is defined as a set of perceivable attributes expressed in a user-friendly language with parameters that may be subjective or objective. Objective parameters are those related to a particular service and are measurable and verifiable. Subjective parameters are those based on the opinions of the end-users. We believe that quality of service should become an integral part of multimedia database systems and users should be able to query by requiring a quality of service from the system. The specification and enforcement of QoS presents an interesting challenge in multimedia systems development. A deal of effort has been done on QoS specification and control at the system and the network levels, but less work has been done at the application/user level. In this paper, we propose a language, in the style of constraint database languages, for formal specification of QoS constraints. The satisfaction by the system of the user quality requirements can be viewed as a constraint satisfaction problem, and the negotiation can be viewed as constraints relaxation. We believe this paper represents a first step towards the development of a formal framework for quality of service management in video databases.

Keywords: *Quality of Service, Multimedia Presentations, Video Databases, QoS Parameters, QoS Specification, QoS Mapping, Constraint-Based Query Languages, Constraint Satisfaction, Constraint Optimization, Reactive Systems.*

1 Introduction

There is a qualitative difference between time-based media and the forms of data traditionally stored in database systems. Time-based media, including digital video and digital audio, music and animation, involve notions of data flow, timing, presentation, etc. These

notions are foreign to conventional database management systems.

Since the usefulness of time-based presentations depends on the accuracy of both timing and data, computing the result of a query in a video[1] database is not only a question of correctness but also of *quality*. Where database design has traditionally been concerned with the delivery of correct results with acceptable delay, multimedia systems present a new challenge: to deliver results with *acceptable quality* in real-time. But how accurate must be a presentation to be acceptable, and how can we guarantee that a presentation achieves that accuracy?

Typical application QoS parameters for images and video include image size, frame rate, startup delay, reliability, etc. The application QoS profile can also include subjective factors such as the degree of importance of the information to the user and the overall cost-quality metric that the user desires. Network QoS parameters include bandwidth, delay, jitter and loss rate. End-system parameters include CPU load, utilization, buffering mechanisms and storage related parameters. Users express dynamic preferences for media quality through benefit functions, e.g., (1) frame rate benefit function which indicates that beyond a threshold frame rate, there is no additional benefit, (2) synchronization benefit function which indicates that the benefit is high only if the audio/video synchronization skew is low.

One particular problem that has been proven to be challenging to solve involves the specification of quality of service. "The human user of a multimedia application is the starting point for overall QoS considerations". In the end, the users of applications are interested in the level of quality of service being delivered. Consequently, quality of service must be considered from the user's perspective, based on the user's expectations associated with applications. In other words, quality of service specifications must be application-level expectations, as opposed to low-level resources reservations.

We partition the QoS parameters into two subsets, namely *application-dependent* parameters and *application-independent* parameters. For example, most electronic commerce applications require multi-media presentation to the customer from the venders, and then the quality of audio and video is important in addition to images, text and numbers. Application parameters describe requirements for application services and are specified in terms of media quality and media relations. Media quality includes source/destination characteristics such as media data unit rate and transmission characteristics such as response time. Media relations specify relationships among media, such as media conversion and interstream synchronization. Researchers have yet to determine the best set of QoS parameters for multimedia systems. Table 1 shows some common QoS parameters in the multimedia community (mainly for video).

Moreover, user demands can be flexible. For example, some users accept only high quality video, while others are satisfied with lower quality when the system capacity cannot

[1] More generally multimedia.

Type	QoS Parameter
Application-Dependent	*Frame Width, Frame Height* *Color Resolution, Time Guarantee* *Space Guarantee, Resource Requirements* ...
Application-Independent	*Delay, Jitter, Reliability* *Throughput, Bandwidth, Packet Loss* *Speed of the Network, Network Topology* ...

Table 1: Examples of Possible Quality of Service Parameters

accommodate them otherwise. Some users allow service degradation as long as specified and agreed upon minimum quality is guaranteed. The system should be adaptable to accommodate various user's QoS requirements.
A deal of work has been done on QoS specification and handling at the system and the network levels, but less work has been done at the application/user level. This paper defines a framework for user-oriented QoS specification and enforcement. The definitions are intended to be general enough to apply to presentations in any multimedia system. We would like to be able to endow multimedia systems with capabilities that will make them able to decide whether a QoS specification is satisfiable or not.

A large number of applications will benefit from video management systems incorporating QoS management. Applications include, for example, monitoring of intensive care units, video on demand, virtual reality, Internet video, video e-mail, virtual cinema, video games, and (distributed) simulation.Though only a partial list, these advanced applications need new techniques and tools for managing QoS issues related to time-based media and mainly video data.

Paper outline: This paper is organized as follows.
This paper is organized as follows. Section 2 summarizes the contributions of this work. Section 3 discusses related work. It gives a brief summary of some of the approaches proposed to tackle the problem of QoS in multimedia databases. Section 4 gives an abstract architecture of a video management system incorporating a QoS manager. Section 5 gives the kind of metadata required for video databases and an appropriate data model for representing this metadata. Section 6 deals with low-level constraints mapping. In Section 7 we provide a logical framework for negotiation. That is a calculus for computing best qualifiers for user-specified quality parameters. Section 8 describes the on-going implementation. We conclude in Section 9 by anticipating on the necessary extensions.

2 Contributions

This paper advocates the use of constraint-based rule language for specifying and reasoning on application-dependent quality of service parameters in video databases. The framework presented here integrates techniques developed in constraint databases and constraint (logic) programming (CLP). The main contributions of the paper are the following:

- We propose a query language, based on a CLP-scheme, for video databases which accommodates QoS parameters (mainly presentation parameters). As in [21], we consider queries as composed of two parts: a *content* part and a *view* part. The *content* part specifies conditions that video sequences should satisfy to be answers to the query. The *view* part specifies constraints for a desired presentation of the outputs.

- We present a terminating procedure, called elaboration, that allows to derive implicit constraints from explicit ones stated by the user. The complete set of constraints will be used to build a presentation schedule and a retrieval schedule.

- When no retrieval schedule can be found to satisfy a presentation schedule, thus a user's request, then some quality parameters have to be relaxed. We show that a framework based on preference logic programming constitutes a nice basis to achieve the goal of what parameters to relax in case of many choices.

Our formulation of quality of service and the problem of its satisfaction by a query offers the benefits of having a simple declarative semantics, providing modularity, and being amenable to an efficient implementation. Many of quality of service aspects have been considered in previous work, and one of our goals is also to unify ideas, provide a more formal foundation, and express these aspects in a way suitable for reasoning. To the best of our knowledge, this is the first logical framework combining techniques from constraint databases and constraint (logic) programming for specifying and handling quality of service in video databases.

Although in the basic form that we give here, the formalism does not account for all aspects of quality of service in video databases, it constitutes a kernel to be extended. Showing how we can formally specify and reason about quality of service is useful and significant.

3 Related Work

Our work relates to several fields of research regarding support of quality of service in multimedia databases. We briefly discuss the relationship to time-based media (mainly video) presentation, quality of service mapping, and specification of QoS in multimedia databases. This section is intended to be illustrative. We apologize if we left out other relevant references.

- **Time-based media presentation.** In [25], the authors proposed an extension of existing object-oriented database techniques to include mechanisms for video presentation. Compressed video data streams are segmented and stored as sets of video objects coupled with specified synchronization requirements. The only quality problem considered in that paper is the one concerning synchronization between the decoder and the viewer of frames. This is done by means of buffer management. User/application QoS is not addressed. [17] proposed a graph data model for the specification of multimedia presentations. Each node of a presentation graph represents a media stream. Edges depict sequential or concurrent playout of streams during the presentation. The paper assumes that multimedia presentations are created and stored in the form of multimedia presentation graphs, which can be viewed as high level abstractions for multimedia presentations. The emphasize in that paper is on finding appropriate language constructs to create and manipulate presentation graphs rather than constraining presentation during database querying. Operators like *Next*, *Connected*, and *Until* and path formulas are used for querying presentations. Hence, augmenting a database by a presentation base could help users to efficiently locate the desired information in the database. Note that with this approach, the quality parameters attached to graphs are statically checked. In contrast, we allow a dynamic specification and checking of quality parameters. Another problem with this approach is that it is not clear which presentations to store in the case of very large databases and how retrieval of presentations can be combined with data retrieval. [24] examined the querying requirements of large libraries of multimedia presentations. The authors proposed an integrated composition and query capability to permit the reuse of multimedia objects, presentations and presentation segments. Here the quality of service is seen as the benefit gained in retrieving and reusing components of presentations. Again with this approach users cannot dynamically specify quality of service parameters as they are already integrated and compiled in pre-defined presentations. [21] defined a methodology for QoS specification regarding presentation. In the proposed framework, a user may control a player's view parameters such as window size and playback rate, as well as quality parameters such as spatial and temporal resolution. When a user chooses to begin a presentation, the player needs to verify that a presentation plan consisting of real-time tasks will satisfy the QoS specification.

  The QoS constraints are specified using the Z specification language [20] which is not intended to be manipulated by end-users. In the framework we propose the constraint part of our queries can be translated into the specifications of [21] and then their framework can be applied to the specifications .

- **QoS Mapping.** In order to guarantee the fulfillment of user/application requirements, a mapping onto the involved operating system and network resources has to be performed. QoS mapping is regarded as the process of translating QoS-parameter bounds from layer to layer, and finally, to resources, e.g., buffers. [14] presented an integrated view of translation and admission control relations for MPEG-video streams between a multimedia distributed application such as video-on-demand, and its underlying system resource, CPU. [15] proposed guidelines as to how QoS-parameters

are affected by stimuli. [5] presented a set of translations of QoS parameters from the application's point of view into transport and operating system QoS parameters. The mapping is a strict one in the sense that no framework is given for negotiation.

These proposals consider mapping as an internal operation carried out by the systems. It is not clear how the operational level of this mapping is specified. In our framework, we specify the mapping by means of declarative rules. Rules' conditions specify the input (source) quality parameters and conclusions specify the target parameters. These target parameters are computed by executing the set of rules on a given quality of service specification expressed by the user.

- **Specification of QoS.** [16] introduced an approach to the formal design and modeling of QoS parameters in multimedia systems. The proposed approach is based on the principle of separation of concerns. The authors use the process algebra based language LOTOS [1] to specify the functional behavior of the system and separately describe the quality of service requirements using an appropriate temporal logic. It is not clear in this paper how the mapping is done and what should be the operational semantics that will be used to perform the satisfiability test of QoS constraints. [6] proposed a conceptual data model for time-based media. The proposed data model includes media descriptors that attach a quality factor to each media object. These media describe the quality of the representation rather than the presentation. This kind of data can be useful for (semantic) query optimization in the case where the user makes reference to these metadata when specifying queries. These informations can be taken into account in our framework as constraints in the view part of queries and they are handled prior to the evaluation of the query (provided that sufficient local informations are available). [22] proposed a model for the specification of user-level QoS preferences in multimedia databases. The quality is defined as a distance measure in multiple quality dimensions, and utility functions are used to capture user QoS preferences in each dimension.

4 A System Architecture Supporting Quality of Service

This section proposes an abstract architecture for a video data management system. This abstract architecture, inspired from the one given in [10], is designed to provide a management system for video data. Figure 1 shows the block level architecture of a video data management system integrating quality of service management. The function of each of these blocks is described below.

User Interface. The user interface in a video data management system plays a critical role in the overall usability of the system. It allows the specification of queries to the database.

Query Processor. The basic function of the query processor is to generate query execution plans. It performs the transformation between the queries formulated by the user into a data model representation which can be used to locate the data.

The user queries can be specified in a variety of different languages. In this paper we consider queries formulated in a constraint-based query language. This will be detailed in Section 5.2.

Quality Manager. This is the main module we are interested in. This module deals with the enforcement of the quality of service specifications. As we will see, quality of service is specified by means of a set of *quality constraints*, and quality constraint checking is performed by this module based on the algorithm we provide in Section 5.3.

Insertion Module. This module deals with the raw video data as it is being inserted into the video database.

Interactive Video Processor. It is used to browse and play-back retrieved video sequences. A user may control view parameters, such as window size and playback rate, as well as quality parameters such as spatial and temporal resolution. In this case, the quality manager module is expected to guarantee the *satisfaction of quality constraints*.

Database Interface. This module provides the interface between the video processing and video query software, and a database management system. The interface translates the video queries formulated by the user into appropriate queries which constitute inputs to the Query Processor.

Video Representation Parameters Store. This module is the database that stores the video representation parameters. Each unit of video in the database is represented by an instance of the video data model. The exact nature of the data model depends on the type of application and the nature of the queries supported by the video data management system. Examples of parameters are color model, duration, etc.

Content Store. This module is the database that stores data referring to content semantics. This data is also referred to as content-descriptive metadata. It is concerned with relationships of video entities with real-world entities or temporal events, emotions and meaning associated with visual signs and scenes.

Video Raw Data. This is the physical store for the video data. The video data may be in a compressed digital form.

5 QoS Formal Specification

In this Section we first introduce some preliminary definitions and then concentrate on QoS constraints and how they are used in retrieving videos according to a user query augmented with specific presentation requests.

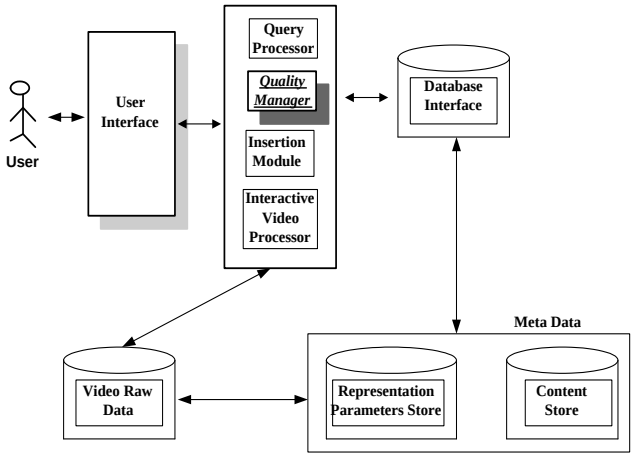

Figure 1: A database infrastructure for video data management.

5.1 QoS Metadata

Meta-information can be used to extend or reduce the search space and the amount of resources required by answers to a given database query. One aspect that is important in our framework is to use knowledge of the underlying media-instance definition.

Metadata is informally defined as "data about other data". It provides concise description of certain data or information collections that may be of interest to the public. In recent years, following the indications of new emerging standards like MPEG-4 and MPEG-7, most natural indexing and retrieval units, different from shots and frames, begin to be investigated.

In our framework, we consider metadata specified in an appropriate data model like the one given in [6]. Examples of metadata are given by:

video$_1$ descriptor {
category = homogeneous,
constant frequency
quality factor = "VHS quality"
duration = 10 minutes
frame rate = 25
frame width = 640
frame height = 480
frame depth = 24
color model = RGB
encoding = YUV 8:2:2, JPEG }

audio$_1$ descriptor {
category = homogeneous,
uniform
quality factor = "CD quality"
duration = 10 minutes
sample rate = 44100
sample size = 16
number of channels = 2
encoding = PCM}

The descriptors also contain information that helps allocate for each stream, a measure of data rate variation (for non-uniform streams), and any parameters needed for decompres-

sion. These information can be used to reduce the number of objects, answers to a query, that must be effectively displayed by maintaining a quality of service.

5.2 QoS query language

In defining the query language, we assume the existence of the unary predicate *display_abstract*. *display_abstract* will be used when the user requires only a browsing version of the frame (i.e., only representative frames are returned). By default complete video sequences are returned. This means that for each video sequence, the system maintains a short version (i.e., representative frames) and a full version. The predicate *display_abstract* is not taken into account during query evaluation, but interpreted by raw video retrieval component.

A query is composed of two parts: *content* part and *view* part. A *content* specification defines a set of logical video sequences to be retrieved in the database. *A view* specification maps content onto a set of physical display regions or parameters by specifying desired constraints. Quality is then a measure of how well an actual presentation of content specification matches the ideal presentation of content on a view.

Definition 1 (Query) *A query is of the form:*

$$\mathsf{Q} \| \mathsf{S}$$

where Q *is the content part of the query, that is the characterization of the set of videos that will be retrieved from the database.* S *is the quality part of the query, specifying a set of constraints that retrieved video sequences should satisfy in order to be displayed. Each video sequence satisfying the content part* Q *will be displayed by maintaining the quality of service specified in the view part* S*.*

Example 1 *The following query:*

$$\begin{aligned}\mathsf{ans}(\mathsf{V}) \leftarrow & \mathsf{video}(\mathsf{V}), \mathsf{category}(\mathsf{V},'\,\mathsf{movie}'), \mathsf{produced\_by}(\mathsf{V},'\,\mathsf{Steven\ Spielberg}') \\ & \mathsf{date}(\mathsf{V},'\,1990') \| \\ & \mathsf{display}(\mathsf{V},\mathsf{W}), \mathsf{coord\_x}(\mathsf{W},\mathsf{X}), \mathsf{coord\_y}(\mathsf{W},\mathsf{Y}), \mathsf{X} \leq 250, \mathsf{Y} \leq 200\end{aligned}$$

retrieves video sequences of type "movie" produced by Steven Spielberg in 1990. Each video sequence, in the answer to this query, will be displayed on the screen in a window 250×200 with a resolution of 320×240.

5.3 Constraints Derivation for QoS enforcement

A presentation schedule is typically created by an author, who specifies what objects should be included in a presentation schedule, when these objects have to be presented to the users and where such objects should appear on the screen.

A presentation schedule in our approach is defined by using a constraint language. Once a presentation schedule has been created, we need to create a retrieval schedule that ensures that the resources needed to deliver the presentation to the client are in fact available. Such resources may include availability (load and buffer) of remote data servers, availability of bandwidth from the network, and the availability of buffer space at the client.

Figure 2 shows the cycle of how presentation schedules and retrieval schedules interact. The user specifies a retrieval query augmented with quality constraints. The augmented query and the meta database are fed to a module called evaluation and elaboration, which retrieves objects answers to the query and derives additional constraints. For each constrained object retrieved from the meta database, the constraint solver solves the attached constraints. A solution is a presentation schedule. Any solution may be picked nondeterministically with a view to creating retrieval schedule for it. If this is possible, then we don't need to go further. If no retrieval schedule can be created for a specified presentation schedule, then we must pick another presentation schedule. This cycle is continued till a presentation schedule is found that has a corresponding retrieval schedule.

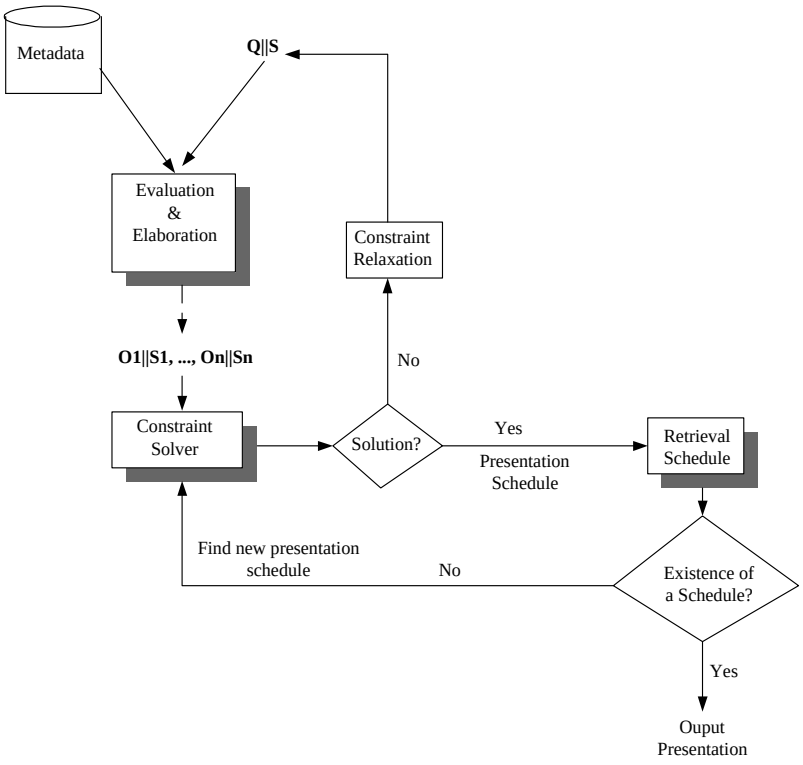

Figure 2: Evaluation schema of Queries involving QoS parameters.

Our idea is to use information from the original query to constrain the search for the objects, and to use intermediate tests to eliminate useless partial solution tuples as soon as possible.

Given a query $Q \| S$, the first step consists in evaluating Q to find the set of objects answers to the query. For each retrieved object O we build two sets $\mathcal{Q}.\mathcal{S}$ where $\mathcal{Q}$ is called the facts set and $\mathcal{S}$ is called the constraints set. $\mathcal{S}$ is the union of S and the constraints attached to O in the meta database.

An *elaboration rule* is an if-then rule that adds constraints to a set of constraints. The following is an example of elaboration rule:

$\mathcal{Q}.\mathcal{S} \rightarrow \mathcal{Q}.\mathcal{S} \cup \{synchronize(X, Y)\}$
if $\mathcal{Q}$ contains video(X), audio(Y), X has_audio Y

Elaboration is applied to a query for the purpose of making implicit constraints explicit, and then accessible to a negotiation algorithm. The propagator applies forward chaining rules to augment $\mathcal{S}$ with constraints that logically follow from $\mathcal{S}$ and $\mathcal{Q}$.

As an example, consider the following query:

$$\begin{array}{l} \mathsf{video}(X), \mathsf{audio}(Y), X \text{ contains } Z, \\ Z \text{ name } "Clinton", X \text{ has\_audio } Y \| \\ \mathsf{display\_time}(t_d), t_d \leq 5, \mathsf{resolution}(X, 'high') \end{array}$$

where t_d stands for display time.

Suppose the evaluation of this query returns the answer $\{X/v, Y/a, Z/o\}$. We have:

$\mathcal{Q} = \{\mathsf{video}(v), \mathsf{audio}(a), v \text{ contains } o, o \text{ name } "Clinton", v \text{ has\_audio } a\}$

$\mathcal{S} = \{\mathsf{display\_time}(t_d), t_d \leq 5, \mathsf{resolution}(v, 'high')\}$

By using the elaboration rule:

$\mathcal{Q}.\mathcal{S} \rightarrow \mathcal{Q}.\mathcal{S} \cup \{synchronize(X, Y)\}$
if $\mathcal{Q}$ contains video(X), audio(Y), X has_audio Y

we can derive the atomic constraint

$$\mathsf{synchronize}(v, a)$$

Now, by using the elaboration rule:

$\mathcal{Q}.\mathcal{S} \rightarrow \mathcal{Q}.\mathcal{S} \cup \{t_{X_{start}} = t_{Y_{start}}, t_{X_{end}} = t_{Y_{end}}\}$
if $\mathcal{S}$ contains synchronize(X, Y)
and $t_{X_{start}}, t_{Y_{start}}$ are variables denoting starting times of X and Y, respectively
and $t_{X_{end}}, t_{Y_{end}}$ are variables denoting ending times of X and Y, respectively

we can derive the atomic constraints $t_{v_{start}} = t_{a_{start}}$ and $t_{v_{end}} = t_{a_{end}}$.

Let t_v be the display time that the video media can provide from $t_{v_{start}}$ and t_a be the display time the audio media can provide from $t_{a_{start}}$. Then, the constraints $t_{v_{start}} + t_d \leq t_{v_{start}} + t_v$ and $t_{a_{start}} + t_d \leq t_{a_{start}} + t_a$ are derivable by using the elaboration rule:

$$\begin{aligned} \mathcal{Q}.\mathcal{S} \quad \rightarrow \quad & \mathcal{Q}.\mathcal{S} \cup \{t_X + t_d \leq t_X + t_v, t_Y + t_d \leq t_Y + t_a\} \\ & \text{if } \mathcal{S} \text{ contains synchronize(X, Y)} \\ & \text{and } t_v \text{ is the display time the video medium can provide from } t_X \\ & \text{and } t_a \text{ is the display time the audio medium can provide from } t_Y \\ & t_X \text{ and } t_Y \text{ being the starting time for displaying } X \text{ and } Y, \text{ respectively} \end{aligned}$$

The augmented set of constraints is then:

$$\begin{aligned} \mathcal{S} = \{ & \mathsf{display\_time}(t_d), t_d \leq 5, \mathsf{resolution}(v, 'high'), \\ & \mathsf{synchronize}(v, a), t_{v_{start}} = t_{a_{start}}, \\ & t_{v_{end}} = t_{a_{end}}, t_{v_{start}} + t_d \leq t_{v_{start}} + t_v, \\ & t_{a_{start}} + t_d \leq t_{a_{start}} + t_a \} \end{aligned}$$

The final set of constraints (called complete set) is the one to which no elaboration rule applies. From the complete set, one derives presentation and retrieval schedules. The problem of scheduling a set of tasks with time and resource constraints is known to be NP-complete [18]. Effective heuristic algorithms exist for this problem [26] which are sensitive to the uncertainty in task completion times.

6 Low-level Constraints

So far, the quality manager dealt only with local constraints, that is presentation constraints or representation constraints, but not constraints (e.g., Jitter, Buffer size) regarding the behaviour or the state of required resources for satisfying quality of service specification. Our extension to such "low-level" constraints is based on the notion of Universal Attachment.

Universal attachment [19] is a domain-independent mechanism for integrating diverse representation and reasoning methods into hybrid frameworks that contain a subsystem based on deduction over logical formulas. Predicate evaluation allows a deductive system to exploit external evaluation procedures by "attaching" programs to predicate and function symbols in a first-order language. When a symbol with an attachment is encountered during deduction, the attached program is invoked to calculate the appropriate value.

Example 2 *The following quality constraint:*

```
panic :-  video(M),
          frame-rate(M, F),
          F > 30,
          os-buffer-size(Z),
          Z < 6600
```

says that if one wants to display a video sequence with a frame-rate greater than 30 then one needs a buffer size greater than 6600 from the operating system. Here, os-buffer-size is a symbol with an attachment. The attached program will be executed at the OS level considered as an object encapsulating a set of operations and services.

6.1 Using Rules to Map Quality of Service Parameters

The high-performance characteristics of the networks (e.g., FDDI, ATM) enable new multimedia applications for which the standardized protocols and services no longer suffice. Especially in the area of service quality, the new applications need mechanisms to express and communicate their needs. It is quite important to provide mechanisms for QoS management, and in particular for mapping application-level parameters to low-level, communication and operating system, QoS parameters.

On the other hand, from the user's point of view, a large set of parameters is unacceptable and normally useless. Users do not want to specify numerous parameters which are often meaningless to them, such as jitter or cell loss ratio. In addition, they have no need to give exact values for certain parameters. A frame rate of 16 frames per second will look quite similar to a frame rate of 14 fps. Therefore only a small set of meaningful parameters should be offered to the user. As we said in the introduction, one of the most important set of parameters from a user's point of view are *presentation* parameters. Examples of such parameters, together with their possible qualifiers are given in table 2.

Parameter	**Domain**	**Qualifiers**
period	milliseconds	*very fast, fast, normal, slow, very slow*
quality	integer	*very high, high, medium, low, very low*
reliability	percent	*very high, high, medium, low, very low*
delay	milliseconds	*minimal, default*
start offset	milliseconds	*minimal, default*

Table 2: Some User Quality of Service Parameters.

Consider the set of network parameters given in table 3. A possible mapping of the user's quality of service parameters of table 2 to network parameters of table 3 is depicted in figure 3.
This mapping can be specified by using our rules. Each mapping rule has the form:

$$\mathsf{D\_P} \leftarrow \mathsf{S\_P}$$

where S_P is the body of the rule which stands for the input parameters at the application level. D_P is the head of the rule specifying the desired target parameter for a given source

Parameter	Domain
Throughput	*bytes per second*
MTU (Maximum Transfer Unit)	*bytes*
reliability	*percent*
burstiness	*integer*
delay	*milliseconds*
jitter	*milliseconds*

Table 3: Examples of Transport QoS Parameters.

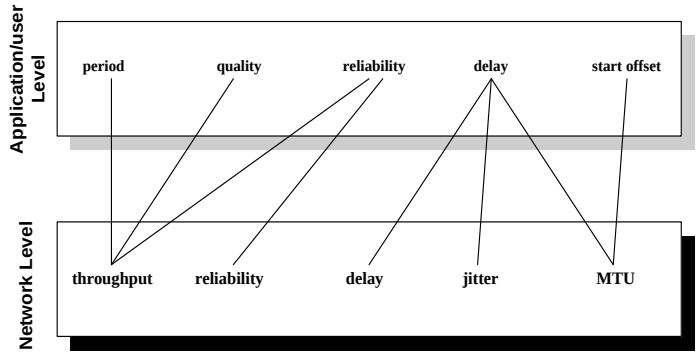

Figure 3: Mapping of Application QoS Parameters to Network QoS Parameters.

parameter. The set of mapping rules should be defined on the basis of consultations with application designers and literature studies. Let us illustrate the dependency between *reliability* and *throughput* [5]. The reliability parameter controls the forward error correction scheme AdFEC (Adaptable Forward Error Correction). AdFEC adds redundancy to the stream to be transmitted such that lost parts of the original stream can be reconstructed. The percentage of parts that are retransmitted is dependent of the qualifier associated with the input parameter *reliability*. For example, if the user asks for a *low* reliability, then we have to use the AdFEC type *FEC_2_1* with a redundancy equal to *33%*. This can be captured by the following simple rule:

$$\mathsf{throughput(redundancy}\rightarrow 33, \mathsf{AdFEC\_Type}\rightarrow \text{'FEC\_2\_1'}) \leftarrow \mathsf{reliability(qualifier}\rightarrow \mathsf{X}), \mathsf{X} = \text{'low'}$$

Note that the syntax of this rule is a slight modification of the one used so far. This is in order to make things more explicit by making use of explicit label names (e.g., qualifier).

7 Negotiation

A considerable research work has addressed the issue of using resource management techniques to prevent fluctuations, i.e., performance variations. Typically, prior to the evaluation of a query, the QoS parameters are mapped into System and Network parameters. In general, this mapping process must result in a set of system resources allocated to the user's demand. A QoS negotiation process and an underlying QoS management infrastructure assure that all system components can provide their share of the requested resources during the session's lifetime. It was shown that this method works reasonably well in homogeneous environments where resources allocation is straightforward and application sessions are not too complex. In many situations, however, this approach is too simple. Multimedia systems may be unable to sustain negotiated levels of QoS. Mechanisms should be developed in a way that allows systems to sustain resource fluctuations within certain limitations. This may be achieved by mechanisms that gracefully adapt to unexpected resource fluctuations.

In what follows, we provide an abstract, logical framework for dealing with resource fluctuations. We call the process negotiation. We show how a framework developed in Logic Programming with Preferences and Constraints for performing optimization and relaxation in a logically principled manner can be used to perform negotiation.

In particular, we consider relaxation and optimization for negotiation. Optimization and relaxation (see, e.g., [2, 9]) are two important operations that naturally arise in many applications involving constraints, e.g., engineering design, scheduling, decision support, etc. In optimization, we are interested in finding the optimal (i.e., best) solutions to a set of constraints with respect to an objective function. In querying video databases, an optimal solution, with respect to the specified quality of service, may be difficult or impossible to obtain, and hence we may be interested in finding suboptimal solutions, by either relaxing the constraints or relaxing the objective function.

First, we introduce some notions and results from Logic Programming with Preferences and Constraints that are necessary for our application (see, e.g., [8] for further details).

7.1 The PLP Framework

A preference logic program (PLP) may be thought of as containing two parts: a **first-order theory** and an **arbiter**. The first-order theory consists of clauses each of which can have one of two forms:

1. $H \leftarrow B_1, \ldots, B_n$, $(n \geq 0)$, i.e., *definite clauses*. Each B_i is of the form $p(\bar{t})$ where p is a predicate and $\bar{t}$ is a sequence of terms. In general, some of the $B_i s$ could be constraints as in CLP or CDB [12, 3, 11, 13].

2. $H \rightarrow C_1, \ldots, C_l \| B_1, \ldots, B_m$, $(l, m \geq 0)$, i.e., *optimization clauses*. $C_1, \ldots, C_l$ are constraints as in CLP that must be satisfied for this clause to be applicable to a goal; they must be read as antecedents of the implication. The variables that appear

only on the RHS of the $\rightarrow$ clause are existentially quantified. The intended meaning of this clause is that the set of solutions to the head is some subset of the set of solutions to the body.

Moreover, the predicate symbols can be partitioned into three disjoint sets depending on the kinds of clauses used to define them:

- *C-predicates* appear only in the heads of definite clauses and the bodies of these clauses contain only *C-predicates* (*C* stands for *core*).
- *O-predicates* appear in the heads of only optimization clauses (*O* stands for *optimization*). For each ground instance of an optimization clause, the instance of the *O-predicate* at the head is the candidate for the optimal solution provided the corresponding instance of the body of the clause is true. The constraints that appear before the $\|$ in the body of an optimization clause are referred to as the *guard* and must be satisfied in order for the head H to be reduced.
- *D-predicates* appear in the heads of only *definite* clauses and any one goal in the body of any of these clauses is either an *O-predicate* or a *D-predicate* (*D* stands for *derived from O-predicates*).

The **arbiter** part of a preference logic program, which specifies the **optimization criterion** for the *O-predicates*, has clauses of the form:

$$p(\bar{t}) \preceq p(\bar{u}) \leftarrow L_1, \ldots, L_n \qquad (n \geq 0)$$

where p is an *O-predicate* and each L_i is an atom whose head is a *C-predicate* or a constraint as in CLP. In essence this form of the arbiter states that $p(\bar{t})$ *is less preferred than* $p(\bar{u})$ if $L_1, \ldots, L_n$.

A preference logic program is a triple of the form $(\mathcal{T}_C, \mathcal{T}_O, \mathcal{A})$ where $\mathcal{T}_C$ is made up of the definition of the *C-predicates*, $\mathcal{T}_O$ consists of the definitions of the *O-predicates* and *D-predicates*, and $\mathcal{A}$ consists of the arbiter clauses in the program.

Note that the C-predicates are different from the D-predicates because the D-predicates are eventually defined in terms of O-predicates. This, however, is not the case with C-predicates because they are defined in terms of other C-predicates only. one can think of the C-predicates as specifying the arbitrary constraints that are to be satisfied by any potentiel solution. The D-predicates on the other hand, are defined in terms of O-predicates.

The pre-interpretation I of interest to preference logic programs interprets functions such as + over the appropriate domain (as in CLP) and leaves all other function symbols uninterpreted (as in Herbrand interpretations).

7.2 Performing Negotiation by Constraints Relaxation

Consider the following preference program of a quality manager:

$$P = (\mathcal{T}_C, \mathcal{T}_O, \mathcal{A})$$

with

$\mathcal{T}_C = \{\mathsf{reliability('very\ high')} \leftarrow \mathsf{throughput(X)}, \mathsf{X} < 100, \mathsf{X} > 66,$
$\mathsf{reliability('high')} \leftarrow \mathsf{throughput(X)}, \mathsf{X} < 66\}$

$\mathcal{T}_O = \{\mathsf{u{-}reliability(X)} \rightarrow \mathsf{reliability(X)}\}$

and

$\mathcal{A} = \{\mathsf{u{-}reliability('medium')} \preceq \mathsf{u{-}reliability('high')} \leftarrow .,$
$\mathsf{u{-}reliability('high')} \preceq \mathsf{u{-}reliability('very\ high')} \leftarrow .,$
$\mathsf{u{-}reliability('medium')} \preceq \mathsf{u{-}reliability('very\ high')} \leftarrow .\}$

throughput in $\mathcal{T}_C$ is an external function that we suppose implemented as a method of the object denoting the network.

Consider a user query $Q \| S$ whose view part S contains the constraint:

u-reliability('medium')

Depending on the current state of the whole system, our objective is to find the best qualifier for the parameter u-reliability. Suppose that the first rule of $\mathcal{T}_C$ triggers for some X between 66 and 100. Then, the fact u-reliability('very high') is generated. Now, the rules of the arbiter part of our program can be applied to select the best solution. Hence, given the two facts u-reliability('medium') and u-reliability('very high'), the arbiter rule u-reliability('medium') $\preceq$ u-reliability('very high')$\leftarrow$. selects u-reliability('very high') as the best solution.

The optimization rules are useful mainly when several qualifiers are computed for a given parameter. In this case, they play a role of a selector.

Rules can be written so that they allow negotiation in both directions. That is, they can express the fact that when resources are scarce, the resource demand must be degraded. On the contrary, the resource demand needs to be increased later on when resources are abundant.

This process of computing the best qualifier for a quality parameter is done for each parameter involved in the view part of a user-query. Let S be the user-specified quality part and S' the one computed by the preference program. To evaluate the corresponding query over the database, we need to check whether S is subsumed by S'. This is because we assumed the S is the minimum QoS required by the user.

7.3 Quality Subsumption

We propose a framework for representing hierarchically structured quality parameters. A hierarchically structured quality parameter is a strict order set of elements.

Definition 2 (Quality Parameter) *A* Quality Parameter *is a chain* $(\mathcal{P}, \succ)$*, where* $\mathcal{P}$ *is a set of qualifiers and* $\succ$ *is a strict ordering.*

Example 3 *Let us consider the reliability quality parameter. The possible qualifiers for this parameter are* Very High, High, Medium, Low *and* Very Low. *We have the following total ordering between these elements:* Very High $\succ$ High $\succ$ Medium $\succ$ Low $\succ$ Very Low.

In what follows, we consider each quality parameter P_q is associated with a unary predicate P, where $P(x)$ holds for all elements x belonging to the ordered set defined by P_q.

Definition 3 (Ordering on Quality Parameter Predicates) *Let* P_q *be a quality parameter predicate and* x *and* y *be two elements of the chain defined by* P_q*.* $P(x) \succeq P(y)$ *if and only if* $x \succ y$ *in the chain associated with* P_q*.*

We define a substitution as a mapping from variables to variables and constants, which is extended to be the identity on constants and generalized to free tuples in a natural fashion.

We extend substitution to abstract elements of chains as follows: let c and c' be two constants of a given chain. Then c could substitute c' if and only if $c \succ c'$. The extended substitution yields a generalized mapping.

Definition 4 (Quality Subsumption) *Let* S *and* S' *be two quality specifications. We say that* S' subsumes S *(and we write* $S \subseteq S'$*) if there is a generalized mapping from* S' *to* S*.*

7.4 Algorithm for Query Evaluation

In this section, we give an algorithm for query evaluation in video databases in the presence of resource fluctuations.

Algorithm 1 ($\mathcal{QE}$)

Require: a query $Q \| S$ (Q is the content part and S is the quality part)
// S is the minimum QoS required by th user
Ensure: evaluate Q by maintaining S satisfiable.
1: $DisplayedSet \leftarrow \emptyset$
// *computeQoS* Computes new quality parameters S' from S and the system state
// This means performing negotiation which consists in
// computing the new qualifiers for the parameters in S
// *computeQoS* stands for the preference program of a quality manager
2: $computeQoS(S', S)$
3: **if** $S' \succeq S$ **then**
4: Have the database to output the first answer x to Q

5: $DisplayedSet \leftarrow DisplayedSet \cup \{x\}$
// Rewrite Q to exclude the already displayed video sequences
6: Rewrite Q as $Q - DisplayedSet$
7: Repeat 2 until no answer satisfies Q
8: **end if**

---

8 Implementation

We started the implementation of the basic functionalities of a prototype for supporting QoS management. The implementation is still undergoing. Here we give the general principles and our approach to this implementation.

The implementation is being carried out on a Sun platform running SunOS 5.6. The architecture consists of several cooperating modules. We have implemented the user interface module using the QT 2.00 C++ GUI application framework. Currently, the interface allows users to specify queries on video objects with varying quality of service parameters. The QoS parameters that we have considered are those given in table 2 as well as video resolution, audio quality and video/audio synchronization. The user's preferences are translated into a QoS specification that is then processed by the Quality Manager module (written in SWI-Prolog[2][23]).
The quality manager is organized as sketched in figure 4. It is implemented as a meta-interpreter written in SWI-Prolog. SWI-Prolog is a Prolog implementation based on a subset of the WAM (Warren Abstract Machine). It was developed as an open Prolog environment, providing a powerful and bi-directional interface to C. The meta-interpreter takes as input the query specified by the user, and three different programs, namely quality program, mapping program, and relaxation program.

9 Conclusion

There is a growing interest in video databases. As video libraries proliferate, aids to browsing and filtering become increasingly important tools for managing such exponentially growing information resources and for dealing with access problems. One of the central problems in the development of robust and scalable systems for manipulating video information[3] lies in supporting *quality of service*. We believe that formal settings will help understanding related problems. This will lead to the development of intelligent systems in order to effectively disseminate, retrieve, correlate and visualize video information.

This paper has described a logical framework for QoS specification in video databases. The framework is equipped with a QoS query language and it allows for reasoning about QoS specifications. Several quality constraints are considered and a precise mapping is defined

[2]http://www.swi-prolog.org//
[3]Multimedia information in general.

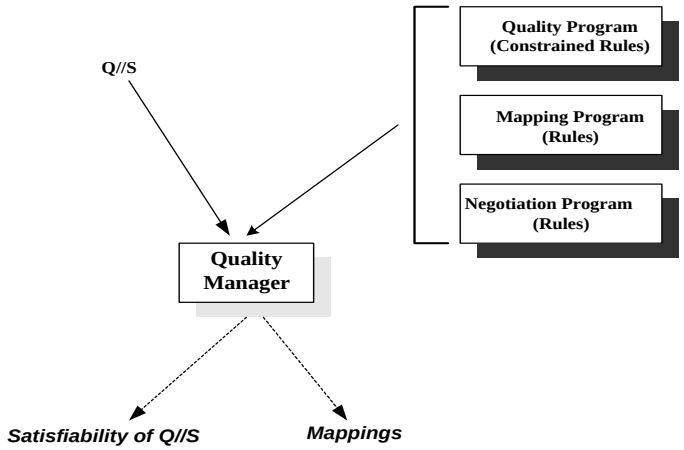

Figure 4: Inputs and outputs of the quality manager

between low-level, i.e. network and OS, constraints and presentation, user-oriented, constraints. Finally, the possibility of resource fluctuation and related constraint relaxation has been discussed and a negotiation algorithm proposed.

There are many interesting directions to pursue:

- An important direction is to extend our framework such that it can accommodate synchronization, concurrency and communication. By considering these aspects at an abstract logical level, it can be possible to predict and check the behavior of the system by reasoning and simulation based on specification, and to give sound reference basis for testing the implementation. This extension will be based on concurrency theory and distributed temporal logics [4]

- While some quality parameters are extensively investigated for low-level layers (i.e., network and operating system), this is not the case for the database level (storage, transaction, etc.). We believe that the support of QoS at the database level requires to devise new query evaluation and optimization strategies.

- Adaptive QoS management may enable a Video DBMS to overcome resource fluctuations by adaptations of media qualities. These adaptations should be optimized towards an efficient utilization of available resources. More specifically, they should yield media objects and composite multimedia presentations with the best possible quality for the given resource availability. Accordingly, the best fitting adaptation is to be computed among the potentially large number of possible adaptations. This is because multimedia objects generally offer multiple parameters that may be adapted. Each of these parameters may have a large number of potential values. For adaptations of composite multimedia presentations, the combinatorial explosion problem is even larger. So, what are the algorithmical solutions for adaptation processing that has the general goal to compute corrective action sets to achieve the optimal adaptation?

- A real-time database is a combination of a real-time operating system, a database, a well-defined constrained execution environment, and a well-defined set of transactions. The inherent complexity of continuous time-based media, such as video data, coupled with the need for efficiency and predictability in real-time systems creates unique and interesting challenges. We believe that even protocols devised for traditional real-time databases need to be modified or redesigned in order to function well in video data environment.

References

[1] T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. *Computer Networks and ISDN Systems, North-Holland*, 14(1), 1988.

[2] A. Brown, S. Mantha, and T. Wakayama. A Logical Reconstruction of Constraint Relaxation Hierarchies in Logic Programming. In *Proceedings of the 7th International Symposium on Methodologies for Intelligent Systems (ISMIS'93), Trondheim, Norway LNAI 689*, pages 362–374, 1993.

[3] Jacques Cohen. Constraint Logic Programming Languages. *Communications of the ACM*, 33(7):52–68, july 1990.

[4] Hans-Dieter Ehrich, Carlos Caleiro, Amilcar Sernadas, and Grit Denker. Logics for Secifying Concurrent Information Systems. In *Logics for Databases and Information Systems, Jan Chomicki and Gunter Saake Eds.*, pages 167–198. Kluwer Academic Publishers, 1998.

[5] S. Fisher and R. Keller. Quality of Service Mapping in Distributed Multimedia Systems. In *Proceedings of the IEEE International Conference on Multimedia Networking (MMNet'95), Aizu, Japan, M. Ikeda, S. Saito, B. Sarikaya (eds.)*, pages 132–141, 1995.

[6] Simon Gibbs, Christian Breiteneder, and Dennis Tsichritzis. Data Modeling of Time-Based Media. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data (SIGMOD'94), Minneapolis, Minnesota, USA*, pages 91–102. ACM Press, May 1994.

[7] Kannan Govindarajan. *Optimization and Relaxation in Logic Languages*. Phd thesis, Department of Computer Science, SUNY-Buffalo, 1997.

[8] Kannan Govindarajan, Bharat Jayaraman, and Surya Mantha. Preference Logic Programming. In *Proceedings of the Twelfth International Conference on Logic Programming (ICLP'95), Tokyo, Japan*, pages 731–745. MIT Press, 1995.

[9] Kannan Govindarajan, Bharat Jayaraman, and Surya Mantha. Optimization and Relaxation in Constraint Logic Language. In *Proceedings of the 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'96)*, pages 91–103, 1996.

[10] Arun Hampapur and Ramesh Jain. Video Data Management Systems: Metadata and Architecture. In Amit Sheth and Wolfgang Klas, editors, *Multimedia Data Management*, pages 245–286. Mc Graw Hill, 1998.

[11] Pascal Van Hentenryck and Vijay Saraswat. Strategic Directions in Constraint Programming. *ACM Comput. Surv.*, 28(4):701–726, Dec. 1996.

[12] J. Jaffar and J.-L. Lassez. Constraint Logic Programming. In *Proceedings of the Fourteenth Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL'87)*, pages 111–119, 1987.

[13] Paris Kanellakis, Gabriel Kuper, and P. Revesz. Constraint Query Languages. *Journal of Computer and System Sciences (JCSS)*, 51(1):26–52, August 1995.

[14] Ki-Wook Kim and K. Nahrstedt. QoS Translation and Admission Control for MPEG Video. In *Proceedings of the 5th IFIP International Workshop on Quality of Service (IWQOS'97), New York*, 1997.

[15] H. Knoche and H. de Meer. Quantitative QoS Mapping: A Unifying Approach. In *Proceedings of the 5th IFIP International Workshop on Quality of Service (IWQOS'97), New York*, pages 347–358, 1997.

[16] Abderrahmane Lakas, Gordon Blair, and Amanda Chetwynd. A Formal Approach to the Design of QoS Parameters in Multimedia Systems. In *Proceedings of the 4th International Workshop on Quality of Service, Paris, France*, 1996.

[17] Taekyong Lee, Lei Sheng, Tolga Bozkaya, Nevzat Hurkan Balkir, Z. Meral Özsoyoglu, and Gultekin Özsoyoglu. Querying Multimedia Presentations Based on Content. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 11(3):361–385, 1999.

[18] J. Lenstra, A. Rinnooy, and P. Brucker. Complexity of Machine Scheduling Problems. *Annals of Discrete Mathematics*, 1, 1977.

[19] Karen L. Myers. Hybrid Reasoning Using Universal Attachment. *Artificial Intelligence*, (67):329–375, 1994.

[20] J. M. Spivey. *The Z Notation: A Reference Manual. Second Edition.* Prentice-Hall International, 1992.

[21] Richard Staehli, Jonathan Walpole, and David Maier. Quality of Service Specification for Multimedia Presentations. *ACM Multimedia Systems*, 3(5/6):251–263, November 1995.

[22] Jonathan Walpole, Charles Krasic, Ling Liu, David Maier, Calton Pu, Dylan McNamee, and David Steere. Quality of Service Semantics for Multimedia Database Systems. *In Database Semantics: Semantic Issues in Multimedia Systems, Edited by Robert Meersman, Zahir Tari and Scott Stevens, Kluwer Academic Publishers*, jan 1999.

[23] Jan Wielmaker. SWI-Prolog 3.3, Reference Manual, http://www.swi.psy.uva.nl/projects/SWI-Prolog. 2000.

[24] Chao-Hui Wu, Renée J. Miller, and Ming T. Liu. Querying Multimedia Presentations. In *Proceedings of the IEEE Conference on Protocols for Multimedia Systems - Multimedia Networking (PROMSMmNET'97)*, pages 64–73, 1997.

[25] Aidong Zhang and Shavinder Multani. Implementation of Video Presentation in Database Systems. In *Proceedings of Storage and Retrieval for Still Image and Video Databases IV*, volume 2670, pages 228–238, San Jose, California, 1996. IS&T SPIE.

[26] W. Zhao and K. Ramamritham. Simple and Integrated Heuristic Algorithms for Scheduling Tasks with Time and Resource Constraints. *Journal of Systems and Software*, 7, 1987.