Specification and Scheduling of Adaptive Multimedia Documents

Stefan Wirag

University of Stuttgart
Institute of Parallel and Distributed High-Performance Systems (IPVR)
Breitwiesenstr. 20-22
D-70565 Stuttgart
e-mail: wirag@informatik.uni-stuttgart.de

Abstract

Multimedia documents are of importance in several application areas, such as education, training, advertising and entertainment. Since multimedia documents may comprise continuous media, such as audio and video, the presentation of those documents may require a significant amount of processing and network resources. The amount of resources available during a presentation depends on the system configuration and the current system load. Hence, it can happen that there are not enough resources to render a multimedia document according to the specification, resulting in a reduced presentation quality, if the presentation is possible at all. To cope with those situations, different versions of the same document can be specified, one for each potential configuration or probable load situation. A better approach is to have only one document that can be adapted to different system configurations and load conditions. To enable this approach, an adaptive document model as well as an adaptive scheduling algorithm are necessary. In this paper, we present the adaptive Tiempo document model, an algorithm to check the consistency of specifications, the concepts of a graphical document editor supporting the model as well as a scheduling algorithm which allows to adapt documents conform to our model in environments with best-effort assignment of resources.

1. Introduction

Due to their expressive power, multimedia documents have become attractive for many application areas, such as education, training, advertising or entertainment. Multimedia documents combine continuous media objects (e.g., video, audio or animation) and discrete media objects (e.g., graphic or text).

In a distributed environment, documents are typically stored on *servers*, from which they are retrieved for presentation. The actual presentation takes place at a *presentation terminal*, such as a workstation, a PC, a Set-Top-Unit, or even a mobile device in future times. When a user initiates the presentation of some document at a terminal, the terminal takes over the responsibility for orchestrating this presentation, i.e. it schedules the access to remote servers, the playout of individual data units, and so forth.

In order to present a multimedia document, a certain amount of resources is needed. For example, processing and buffer resources are needed at the terminal to present the document, while network resources are required to transfer the media objects associated with the document from the server to the terminal. The amount of resources available strongly depends on the system configuration and the current system load.

Let us consider the system configuration first. Clearly, a workstation connected to a high-speed network allows for a higher quality presentation than a PDA linked to a radio network. One approach to overcome this problem of heterogeneity is to have different versions of the same “logical” document, one for each potential configuration. Another approach is to have one document that is able to adapt to the capabilities of the underlying system. We prefer the second approach since it avoids redundancy and does not have to predict numerous configurations. For example, with the second approach a digital library would have to store only one version for each document, without taking into account the capabilities of the terminals potentially used to access the library.

Even if we only consider one type of terminal and one type of network, variations in the system load may cause different amounts of resources to be available. Without resource reservation, the resources available for a presentation may change while the presentation is in progress, i.e., resource shortages cannot be avoided. If the underlying system provides for resource reservation, the resources needed to present a (mono-media) object (e.g., a video clip) can be determined and reserved prior to its presentation. However, reservation in general cannot prevent resource shortages to occur during the presentation of interactive multi-media documents. For interactive documents, it is not feasible or even possible to reserve all resources required to render the entire document in advance. Rather resources are reserved and releases incrementally while the presentation progresses, which again can lead to resource shortages. Consequently, even in the case of resource reservation it may happen that less resources are available than expected.

When a resource shortage occurs, there are basically three ways to react on it. Firstly, the presentation is aborted, which is the only choice if the underlying system is non-adaptive. Secondly, the quality of the presentation is degraded *somehow,* meaning in a system-controlled manner. In this case, the underlying system is adaptive, however the author of the document cannot impact how the quality is degraded. Thirdly, the presentation is adapted in a *user-controlled manner* to the given resource situation. To enable the last alternative, flexible document models are required, which allow to compile different presentations from a given document specification depending on the resource situation.

In the *TIEMPO*[1] project which was supported by the Deutsche Forschungsgemeinsachaft (DFG), a flexible document model [12, 15, 16] was developed. In this model documents are composed of single media objects, such as video, audio or text, and composite media objects, such as pages or scenes. The desired adaptability is achieved by *selection groups* which allow to define alternative media objects or presentation parts representing the same information in different form and *Quality of Service ranges* which allow to specify alternative presentation behavior for media objects. These abstractions can be applied in combination to achieve a high degree of adaptability.

To check if flexible documents are presentable, we have developed a graph-based consistency-checking algorithm, which is used in the graphical Tiempo document editor to detect inconsistent specification parts. The adaptive scheduling algorithm of Tiempo uses the flexibility in documents to adapt the presentation to alternating resource situations. The algorithm schedules a presentation such that not more resources are needed than available and that the available resources are distributed optimal between simultaneously presented media objects. Thus, resource shortages need not result in an uncontrolled reduced presentation quality of Tiempo documents.

The remainder of the paper is structured as follows: In Section 2 we present the concepts of the Tiempo model. Then, we describe the consistency-checking algorithm. In Section 4 a brief overview of the document language the Tiempo editor is based upon is given, before the design concepts of the editor are presented. In Section 6 to 9 we describe our adaptive scheduling algorithm. In Section 10 some performance measurements are presented. Related work concerning is described in Section 11. Finally, we summarize our results.

1. **T**emporal **i**nt**e**grated **m**odel to **p**resent multimedia-**o**bjects

2. The TIEMPO Document Model

2.1 Basis Concepts

Tiempo is an interval-based model, in which documents are composed of single media objects and composite media objects. A media object is modeled by a temporal space, a presentation interval and a projection. The *temporal space (TS)* represents the content and layout information associated with the media object. The *presentation interval* represents the period the media object is presented. The *projection* describes which and how many data units of the TS are presented per second in the presentation interval. The concept of a projection allows to present media objects with other than recording-time properties. Our model knows four types of projections: With the type *align-length* the whole TS is presented in the presentation interval. With the type *hold-data* the presentation interval can have any extent, gaps at the edges of the presentation interval are filled with the last or first data unit of the TS. Applying the type *loop-data,* a presentation interval of any length can be specified. Gaps at the edges of the presentation interval are filled by repeating the already presented part of the TS. With the type *force-end* a presentation interval of any length can be defined, until the TS is large enough to fill the interval. Interaction objects, such as buttons or sliders, have additional *interaction intervals.* Each such interval represents the period in which a particular user-interaction (e.g. click with mouse) is accepted.

A TS consists of a finite time axis on which data units such as video frames, audio samples or pictures are positioned. The arrangement of data units and the extent of a single media objects TS is fixed when its content is generated. To define the temporal layout of a composite media objects TS, presentation intervals of included media objects are arranged by *interval operators* [12]. In Figure 1, the left side shows a simplified composite media object specification and the right side shows how the temporal layout of the presentation is generated by projection of TSs. In this example, the interval operator “before” specifies that the animation should be started 1 second after the video has ended.

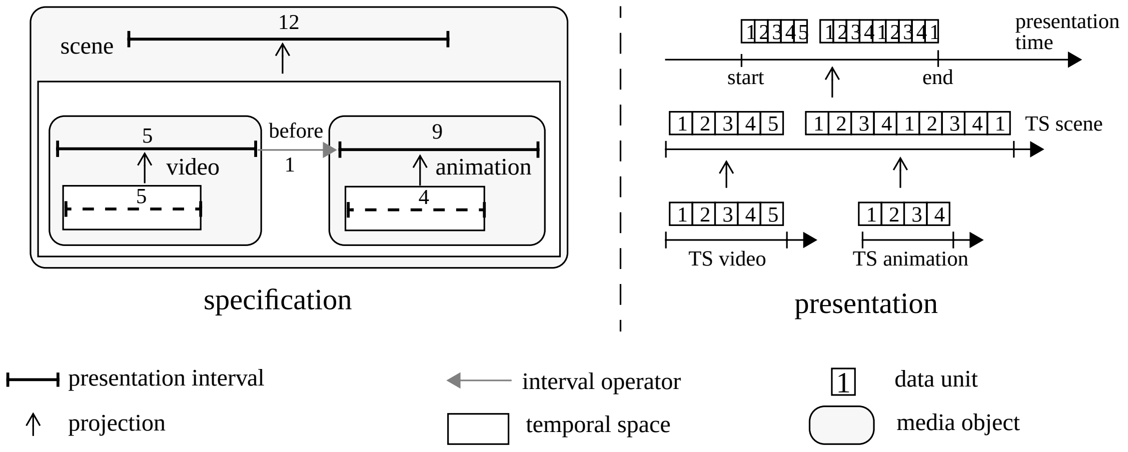

Figure 1: Tiempo specification example

An interaction is described as a reaction-relation between an interaction interval and the affected projections, interaction and presentation intervals. Assigned actions can imply the start or end of intervals, they can pause, continue and speed up the playout [13].

2.2 Modeling of Adaptivity

The Tiempo model allows for adaptivity on the object- and attribute-level [15, 16]. Adaptivity on the object level can be specified by so-called *selection groups*. A selection group contains a number of presentations that can be selected alternatively. Whenever a selection group is performed, the underlying system selects and presents exactly one of these alternatives. A presentation alternative can be a single or arbitrarily complex composite media object. Selection groups can be nested to specify selections at various levels of abstraction. Moreover, priorities can be assigned to presentation alternatives to indicate which alternative should be preferred when more than one can be implemented.

Figure 2 shows an abstract representation of a document part with nested selection groups. The outer group includes two presentation alternatives. While the first alternative is a text object, the second one consists of an animation object and another selection group. The inner selection group provides three presentation alternatives, a speech object, a subtitle sequence and an empty object. According to the selection group semantics, the presentation system has the option to present the text or the animation with either the speech sequence, the subtitles or no further explanation.

In the outer selection group, the alternative with the animation has the highest priority (100) and hence should be selected provided the required resources are available. In the inner selection group, the speech object is the preferred alternative (priority 80), followed by the subtitles (priority 60) and the empty object (priority 0).

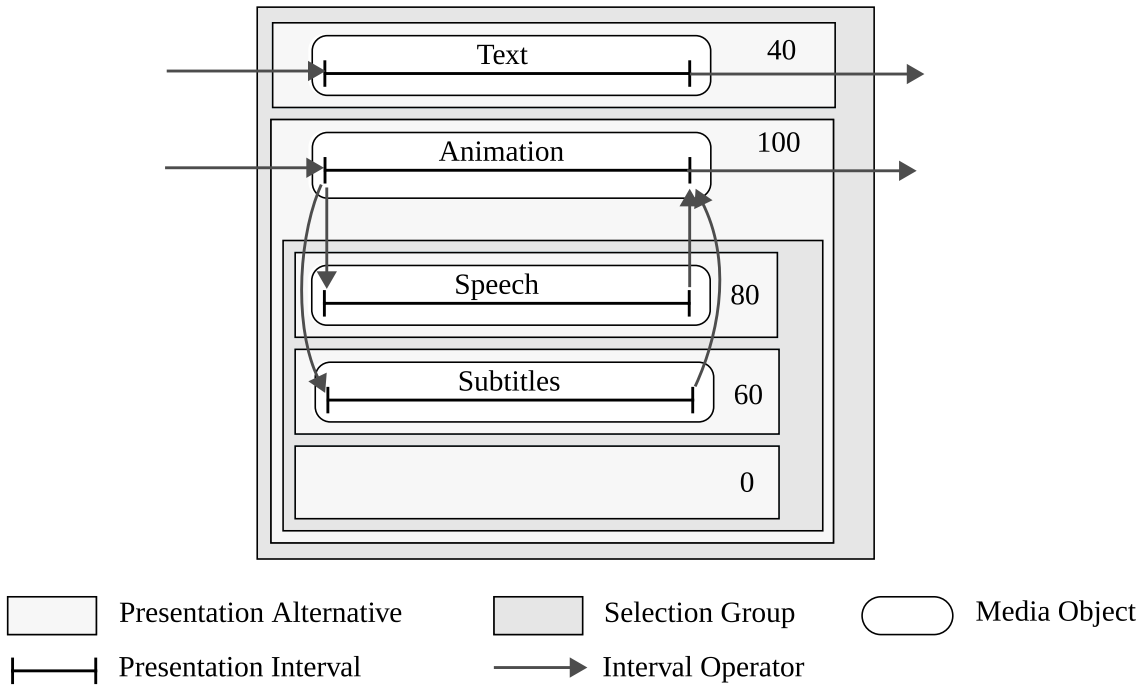

Figure 2: Document specification with selection groups

Adaptivity on the attribute-level is modeled by so-called *Quality of Service (QoS) ranges*. QoS range arguments can be used to specify the extent of presentation and interaction intervals, the presentation speed and the delays implied by interval operators. Based on this information, a presentation system can select extent, speed or delay values within the specified QoS ranges according to the current resource situation. To indicate which values of QoS ranges are preferred, priorities are assigned to the contained values. The priority structure of a QoS range is defined by anchor points. The QoS range on the left side of Figure 3 might define the extent of a presentation interval, which can be between 10 seconds and 55 seconds. In the example, an extent of 10 seconds has the priority 30, an extent of 55 seconds has the priority 70, and an extent of 35 seconds has a priority of 100. With the extents between 10 and 35 linear increasing priorities from 30 to 100 are associated, and with the extents between 35 and 55 linear decreasing priorities from 100 to 70 are associated. Here, the presentation system should implement an extent of 35 seconds for an optimal presentation quality.

[10 : 30, 35 :100, 55 : 70]

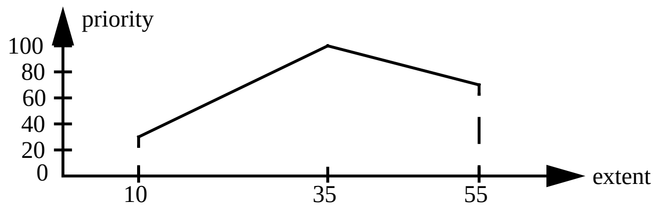

Figure 3: QoS range semantics

3. Consistency Checking

Tiempo specifications may integrate many flexible temporal dependencies. Hence, it is necessary to check if all temporal relations are compatible. Otherwise the document is not presentable.

3.1 Basic Algorithm

To check the consistency of Tiempo specifications an graph-based algorithm is used. The application of a graph-based algorithm is possible as a Tiempo specification can be represented as a directed acyclic graph. Such a graph is called event-graph. An event-graph is derived as follows from a specification.

In an event-graph each interval in a specification is modeled by two nodes representing its start- and end-event. These nodes are related by an edge which represents the delay between the nodes respectively events. A TS is also modeled by two nodes representing its start- and end-event. According to the specified projection, presentation interval nodes and TS nodes are related by appropriate edges. All our projections can be modeled by introducing appropriate edges [14]. Delays implied by interval operators are modeled as edges between presentation interval nodes, interaction interval nodes and TS nodes. Figure 4 shows on the left side a simple document specification and on the right side the corresponding event-graph of the specification.

The edges in an event-graph are labeled with the possible delay values between the nodes. If speed values, interval lengths and delays implied by interval operators are specified as QoS ranges, edges will be labeled with a value range that contains all possible delays. In Tiempo the projections define how fast data units of a TS are presented. As the nodes of the scheduling graph should describe instants of the real time, delay values specified in a TS i have to be divided by the value $v_{e,i}$ that represents the effective speed in the TS. The effective speed $v_{e,i}$ of a TS i is defined recursively by $v_{e,i} = v_i v_{e,j}$ where v_i is the speed defined for the projection from TS i to TS j and $v_{e,j}$ is the effective speed of TS j. The grey rectangles on the right side of Figure 4 show how edges are affected by the different speeds in the TSs.

Figure 5 shows a simplified specification that consists of a multimedia object containing a slide presentation which is accompanied by a speech sequence. If the presentation of the slide ends, the slide presentation continues until the user presses the stop button. The start of the slide presentation is specified flexible. The whole presentation should be rendered with double speed (v = 200). Not specified lengths

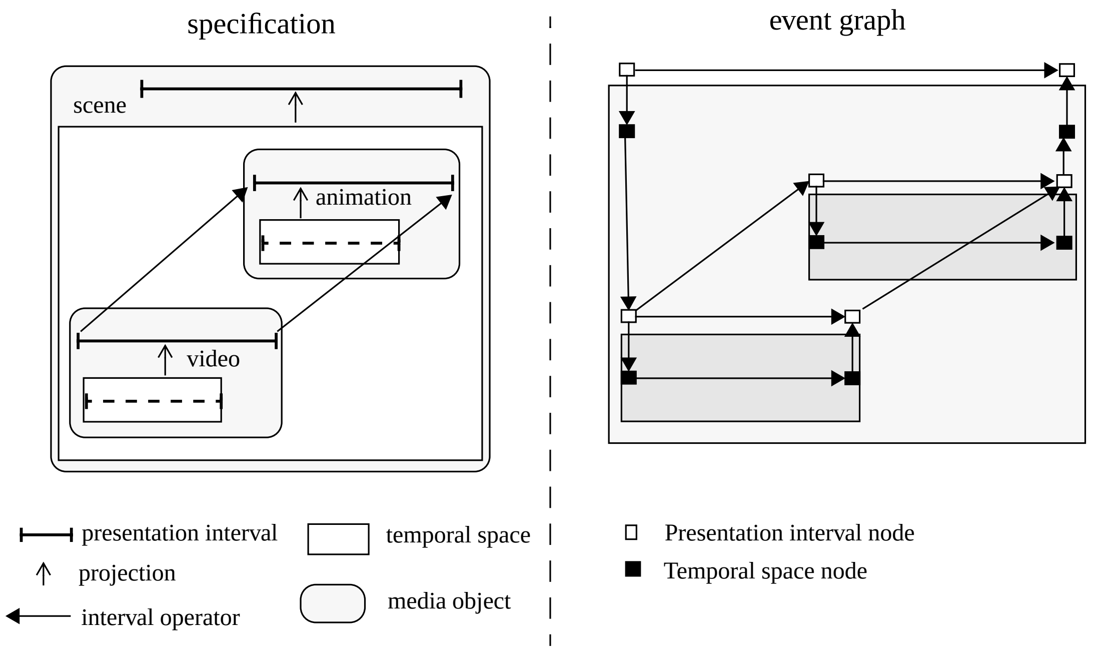

Figure 4: Generation of Event Graphs

and delays are represented by a ‘?’. The left side of Figure 6 shows the event-graph of the example. As can be seen, not specified delays are represented by [0,?], since the delay may lie between zero and infinity. Specified delays are divided by two as the presentation shall be rendered with twice the normal speed.

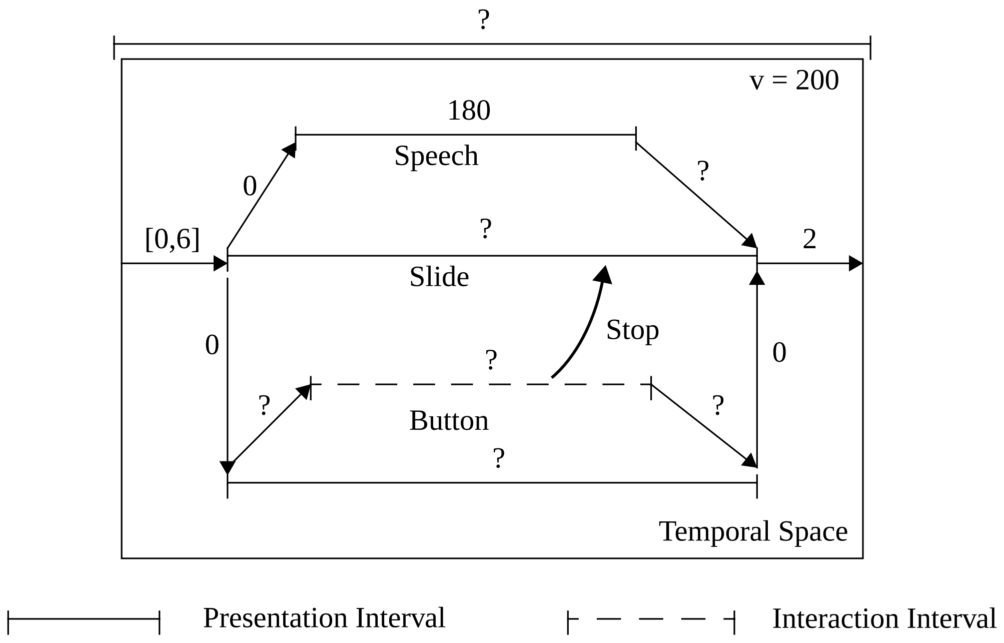

Figure 5: Concistency checking sample specification

With the help of an event-graph it is now possible to check whether all delays represented by edges in the graph can hold simultaneously by application of the algorithm described in [3] as follows.

From an event-graph a so called distance-graph is derived represented as adjacence matrix. In a distance-graph an edge of an event-graph is represented by two edges. A forward edge represents the maximum delay between the nodes and a backward edge represents the minimum delay between the nodes. Further the sign of the value of the backward edge is switched.

On such an adjacency matrix the algorithm of Floyd-Warshall [11] is applied. This algorithm is used to compute the minimal distances between all nodes of a usual graph. Applied on a distance-graph the following is computed:

- Contains the diagonal of the computed matrix negative values is the specification not consistent. This means there exists no combination of edge values so that all minimum and maximum distances are kept.
- Contains the computed matrix only zeros in the diagonal is the specification consistent. This means there exists at least one combination of edge values so that all distances stay within the given intervals. Further on, the computed matrix contains now the possible minimum and maximum distances between all nodes. On the base of this values it is now possible to limit the range of not specified QoS Ranges and we also know now the minimum and maximum distance of all nodes from the start node. Hence, we know in which intervals start- and stop-events of media objects can occur.

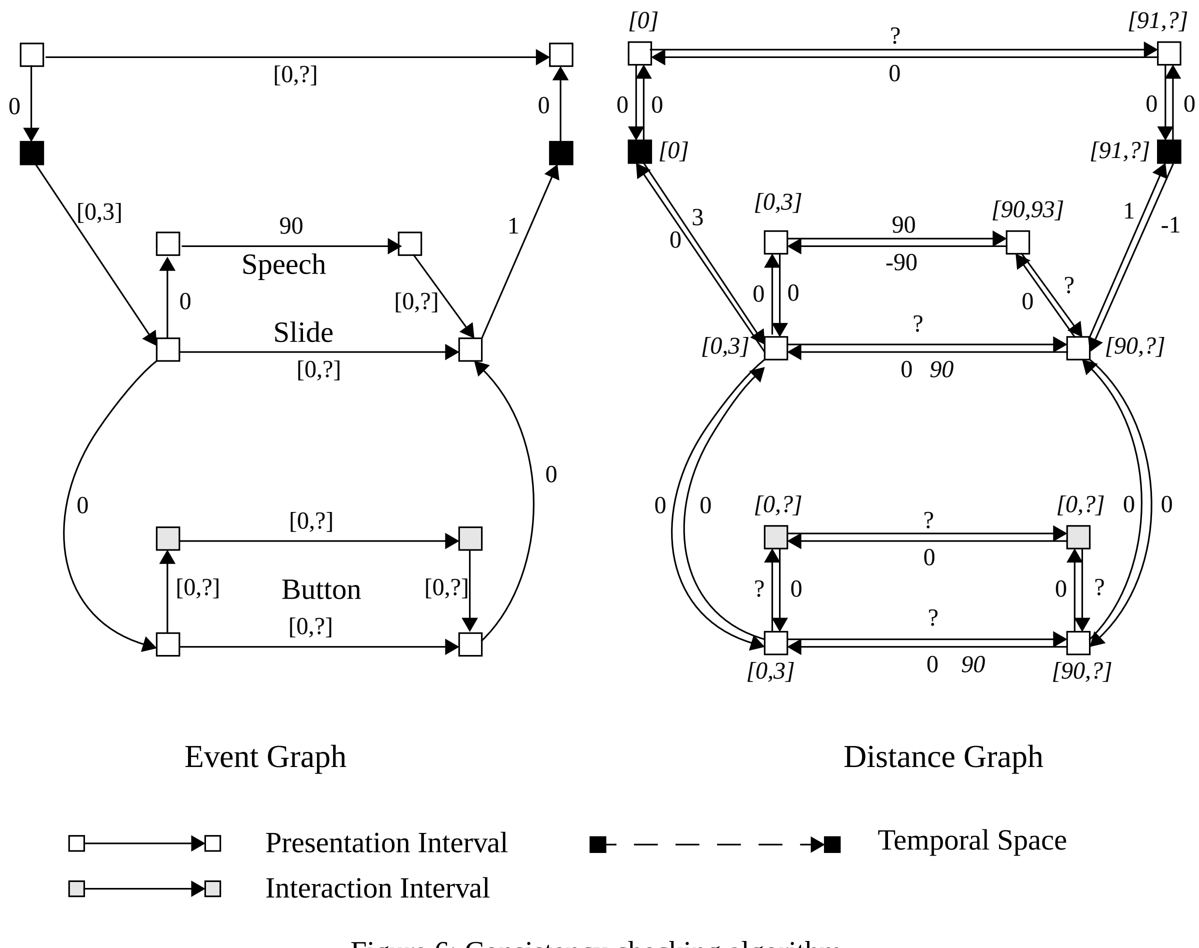

Figure 6: Consistency checking algorithm

If the algorithm is applied on the example in Figure 5, it is shown that the specification is consistent. The distance-graph in Figure 6 shows the computed intervals events associated with the nodes may occur in italic style. These values show that the slide and button presentation can not end earlier than 90 seconds after the start of the presentation. Hence, we can complete their minimum presention duration.

The complexity of this algorithm is determined by the complexity of the Floyd-Warshall Algorithm which is O(n^3) (n is the number of nodes in a graph).

This algorithm can be applied on each media and multimedia object separately. In a hierarchical document specification we can first check the consistence of the objects on the lowest level. Then we can check the objects on the next level and so on. If a specification integrates selection groups, the consistency is checked for each possible combination of presentation alternatives separately.

3.2 Checking Consistency of Interaction

Interaction intervals describe when an interaction can be triggered by the user. Reaction-relations define how media objects should change their behavior when an interaction occurs. In a consistent specification it must be guaranteed that the triggering of an interaction do not cause an inconsistency. The consistency of interactions can be checked as follows.

The consistency of the document is checked without considering interaction effects. Not specified values are completed and QoS Ranges are corrected. Then delays in the event-graph are set as they can be if the interaction would be triggered. On the modified event-graph the described algorithm is applied. On base of the computed minimum and maximum delays between nodes, it can now be checked if the interaction intervals are specified correctly. If causal dependencies of interactions and interaction intervals exist, we check each scenario step by step.

In the example the consistency of the stop-interaction would be checked as follows. The length of the slide presentation is set to [0,?], since the media object will be stopped sometimes by the interaction. Then we compute the distance graph. For the example this distance graph is identical to the distance-graph used to check the consistency without considering the interaction. The computed distance-graph shows that the slide presentation can not stop earlier than 90 seconds after the start of the presentation. Hence, the interaction interval can start not earlier than 90 seconds after the presentation has been started. We can now set the delay between the start of the button presentation interval and the button interaction interval to [90,?].

4. The Tiempo Document Language

Based on the Tiempo model, a document language was developed that supports all features of the temporal model. The developed document language follows the paradigm of object orientation. The language is defined by a set of element types. An element type defines the attributes a document element conform to the type has to have. Also a default-value can be defined for each attribute. To describe the Tiempo model appropriately, the following attribute types are necessary: integer, floating-point numbers, enumeration types, strings, references to objects, attributes and external entities, QoS ranges, and composite attributes. Composite attributes can be compared with struct-constructs in C. These attributes enable a clear representation of complex element properties such as alignment policies. All element types are derived from each other. An element type inherits the attributes of the element types it is derived from but can have additional attributes representing new properties. Multiple inheritance is allowed. Element types can be abstract. Such types can be used to structure the element type hierarchy accordingly. This approach can be compared with the concepts of MHEG [6] or HyTime [4] where document elements are also derived from each other. Tiempo documents are composed by instances of such element types.

Figure 7 shows an example of an inheritance tree that shows the characteristic element types implementing the concepts of the Tiempo model. All element types are derived from the type *TSObject*. The subtree *Media* contains the types that represent media items such as video-clips, speech-sequences, etc. The concrete presentation of media items is defined by viewer that are associated with the media items. The separation of content and presentation functionality has the advantage that the same content can be presented in different form. But viewer can not only present single media, viewer that define the presentation of multiple other viewer are called *MultimediaViewer*. A simple example for such a viewer is a Window. The abstract type *Viewer* has all attributes that are necessary to describe a media object according to the Tiempo model.

The subtree *Relation* contains typs to define dependencies between other elements. This subtree contains types representing the 10 interval operators and the reaction relation. *Reaction* elements contain elements of the types in the *Action* subtree. This subtree contains types representing the interaction effects supported by the Tiempo model.

To describe selection groups, elements of the type *SelectionGroup* are used which contain elements of the type *PresAlternative* representing presentation alternatives.

Composite attributes (*CompositeAttribute*) are also defined within the inheritance tree. Figure 7 shows which composite attributes are necessary to represent alignment policies, projections and intervals.

The Tiempo language can be extended very easily by new kinds of media items. To do this, a new media type describing the media item has to be defined and inserted into the subtree *Media* of the inheritance tree. If no adequate viewer exists also a new viewer object has to be defined and inserted in the subtree *Viewer*.

5. Design of the Tiempo-Editor

The Tiempo-Editor permits the graphical interactive composition of Tiempo-documents. Figure 8 shows the GUI of the editor. The editor integrates the following concepts.

5.1 Views

As it is not possible to display all aspects of a document in a single view, the Tiempo-editor visualizes a document specification in three different views. Each view represents a particular aspect of the specification. A multimedia element is visualized by three views:

- The *Main View* displays media- and multimedia elements contained in a multimedia element by icons. The kind of the icons shows the type of the elements. Temporal relations and reaction relations are also represented by icons. Lines show between which other elements the relation is defined. Selection groups are represented by icons on the left side of the view. With the help of the icons the author can select the displayed presentation alternatives. In Figure 8 the middle window on the left side shows the Main View of a multimedia element.
- The *Temporal View* displays the temporal layout of the multimedia element in detail. Media elements are represented by bars positioned on a time-line. The length of the bars is equal to the presentation duration of the media item. The height of the bar shows the presentation speed of the media item. The position of the bar is given as defined by the specified interval operators. Interval operators are visualized by lines between the bars. If the specification contains QoS Ranges, the Temporal View shows always the layout of the presentation which implements the best quality on attribute level. If the multimedia elements contain selection groups, the Temporal View shows always the presentation alternatives which are currently active in the Main View. In Figure 8 the middle window on the right side shows the Temporal View of the multimedia element on the left side.

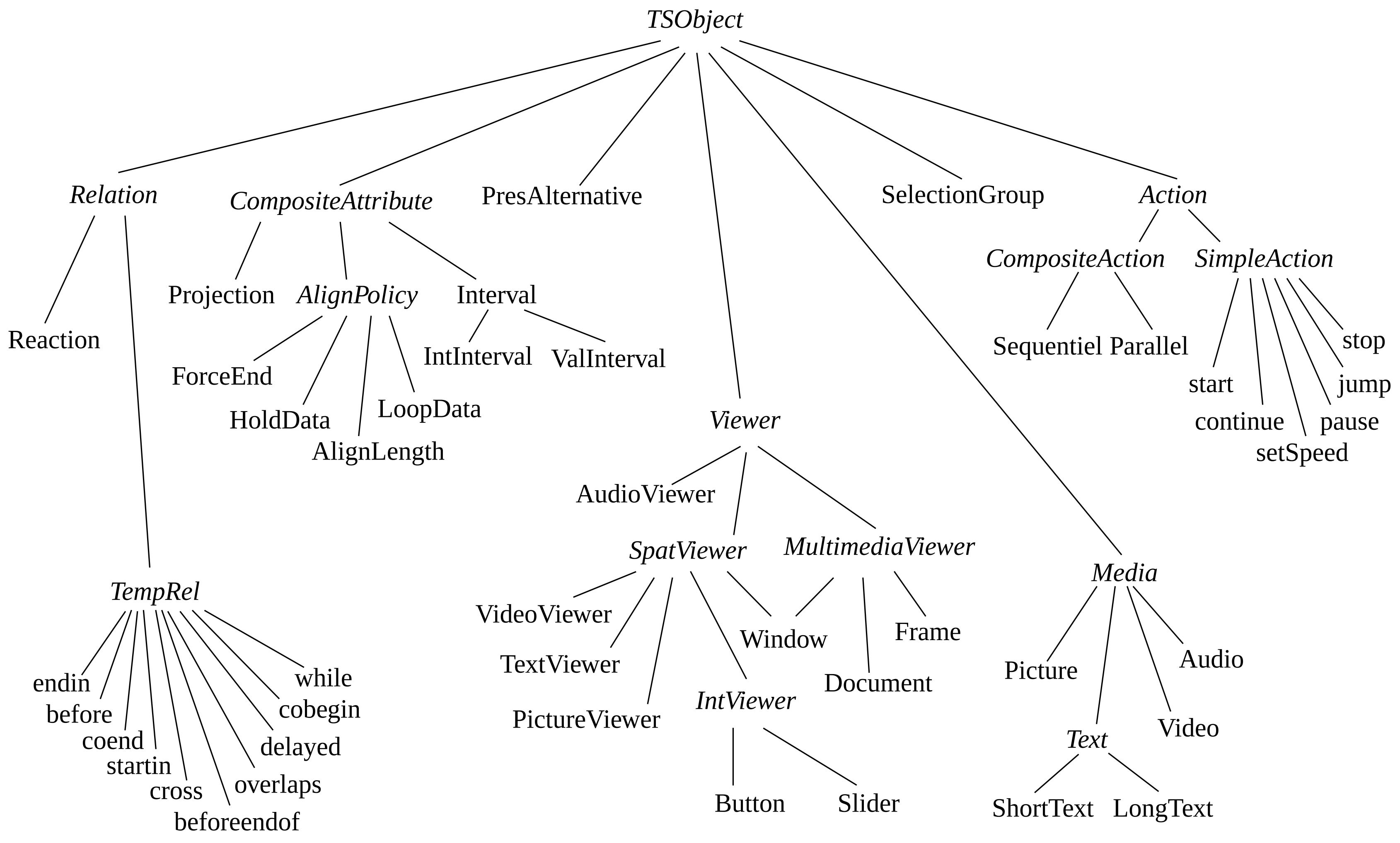

Figure 7: Tiempo Inheritance Tree

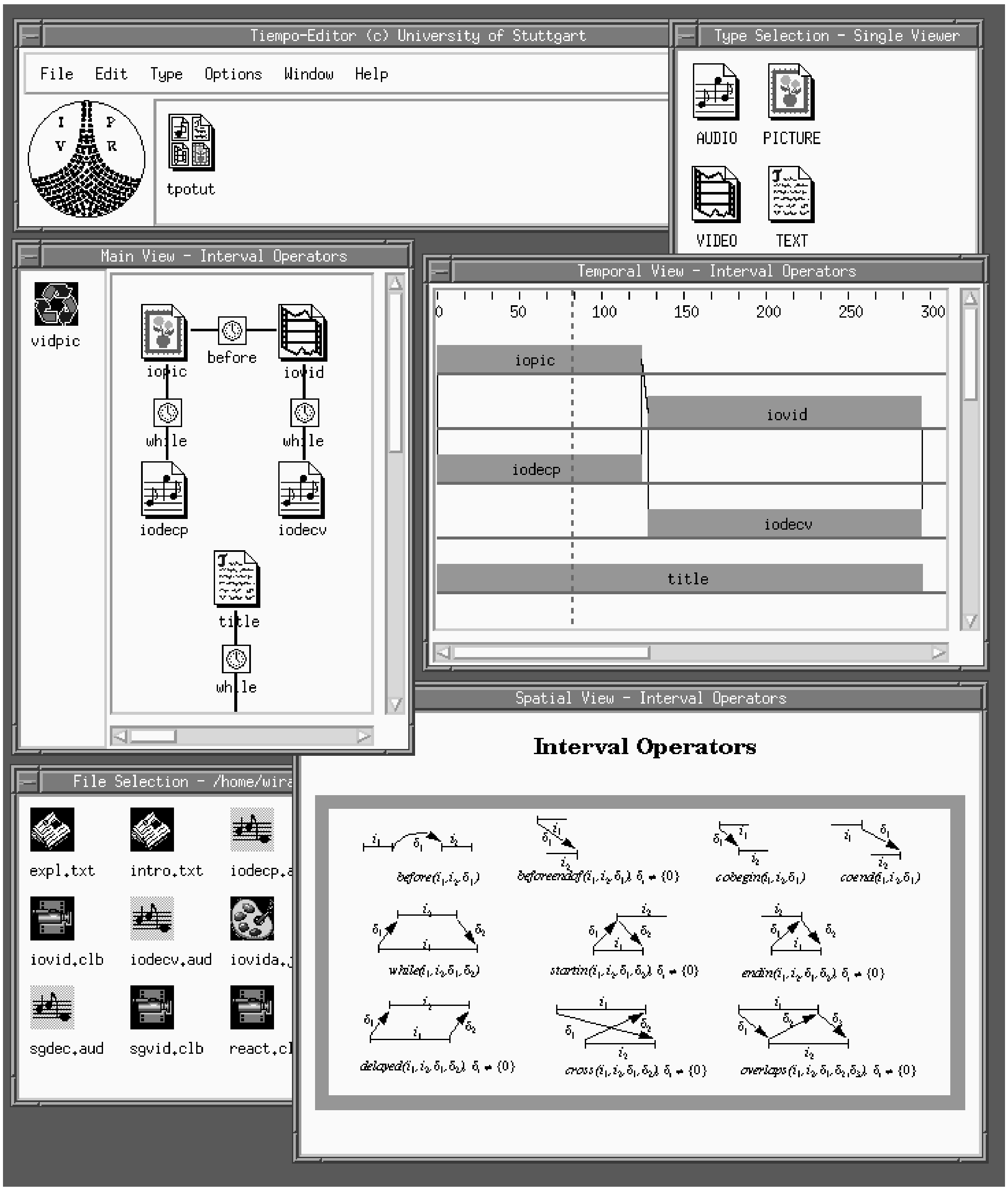
Tiempo-Editor (c) University of Stuttgart
File Edit Type Options Window Help
tpotut
Type Selection - Single Viewer
AUDIO
PICTURE
VIDEO
TEXT
Main View - Interval Operators
vidpic
iopic
before
iovid
while
while
iodecp
iodecv
title
while
Temporal View - Interval Operators
0
50
100
150
200
250
300
iopic
iovid
iodecp
iodecv
title
Spatial View - Interval Operators
Interval Operators
before(i1,i2,δ1)
beforeendof(i1,i2,δ1), δ1 ≠ {0}
cobegin(i1,i2,δ1)
coend(i1,i2,δ1)
while(i1,i2,δ1,δ2)
startin(i1,i2,δ1,δ2), δ1 ≠ {0}
endin(i1,i2,δ1,δ2), δ1 ≠ {0}
delayed(i1,i2,δ1,δ2), δ1 ≠ {0}
cross(i1,i2,δ1,δ2), δ1 ≠ {0}
overlaps(i1,i2,δ1,δ2,δ3), δ1 ≠ {0}
File Selection - /home/wira
expl.txt
intro.txt
iodecp.a
iovid.clb
iodecv.aud
iovida.j
sgdec.aud
sgvid.clb
react.cl

Figure 8: Editor GUI

- The *Spatial View* displays the spatial layout of a multimedia element. Media items are displayed as the are displayed during the presentation with regard to content, size, overlapping and position. As different media items played after each other in a presentation can share the same place, the author can select the displayed instant in the Temporal View. In Figure 8 the window in the lower right corner shows the Spatial View of the sample multimedia element.

With the help of these views temporal and spatial aspects of a specification can be visualized adequately.

5.2 Visualization of Document Elements

To support an easy specification of documents, it is not only necessary to visualize the created documents. It is also necessary to visualize the elements and element types documents are composed of.

Element types of the Tiempo language are visualized in so called type selection panels (TSP) by icons. Each set of element types with similar semantics (e.g. multimedia viewer element types, relation types etc.) is presented in a separate TSP. In Figure 8 the window in the right upper corner shows the TSP of single viewer element types.

Media element types representing media items such as video-clips or music pieces do not contain the data itself. They only contain references to the data. To permit a simple usage of such media items, the editor allows to visualize NFS and Web-server directories by so called File Selection Panels (FSP). A FSP displays the files in the directory according to the type of media they contain by different icons. The editor is capable to automatically recognize and read the attributes of files with JPEG pictures, CellB- and MJPEG videos, SUN audiofiles and HTML-documents. The attribute values are automatically inserted in the media object instances shown in a FSP. In Figure 8 the window in the lower left corner shows a FSP.

Further on, the editor has a clipboard which shows the currently loaded documents as icons. The clipboard shows also specified elements which are currently not part of a document. As can be seen in Figure 8, the clipboard is integrated in the main window of the editor.

5.3 Object-oriented Interaction Mechanisms

The editor integrates object-oriented interaction schemes. Graphical elements such as icons, lines etc. associated with elements have a context menu. Dependent on the element type, the context menus allow operations such as deleting, setting attribute values etc.

When specifying documents, elements are generated by moving the associated type icons with drag&drop into the Main View of the element the new element shall be placed in. When dropping the icon, the new element type instance is generated and the attributes are initialized with default-values. If new instances of interval operators or reaction relations are generated, it is then necessary to create the links to appropriate elements. To generate a new document, a new document instance has to be created by drag&drop in the clipboard. Then its Main View can be opened and other elements can be inserted. To assign a media element to a single viewer element, a media element from a FSP can be dropped onto the viewer element.

In this way, a sketch of a document can be composed quite fast in the Main View. Details can then be specified with the help of context menus or the Temporal and Spatial View. In the Temporal View the presentation speed can be manipulated by squeezing or stretching the bar of an object in vertical dimension. The presentation duration can be manipulated by squeezing or stretching the bar in horizontal dimension. It is also possible to manipulate the position of the bar or the length of lines belonging to interval operators. QoS Range values have to be manipulated textually. In the Spatial View the author can position media items with the mouse. He can manipulate the overlapping and the size.

The editor supports cut, copy and paste operations. During the operation the cutted or copied elements are stored in the clipboard.

The usage of drag&drop techniques and the strong object orientation has the advantage that the author has to know and remember only a few interaction mechanisms.

5.4 Support by Consistency Checking

The editor supports two specification modes. In the first mode a consistency checking is performed after each action of the author. This is done by application of the described consistency checking algorithm whenever the author inserts or removes an element from a document or when she manipulates a temporal attribute. If the action causes the inconsistency of the specification, the author gets an appropriate message. Further, not specified attribute values are set according to the information in distance graphs. The second mode performs no automatic consistency checking. This mode is well suited if bigger modifications in the document are made. The author can switch arbitrarily between the two modes.

6. Adaptive Scheduling Concept

To render a multimedia document, a presentation schedule, which guides the presentation, has to be generated according to the document specification. If documents are interactive, presentation schedules have to be modified appropriately when interactions occur. In our approach presentation schedules are also adapted dynamically to changing resource situations. At each adaptation of the presentation schedule, it is determined which alternatives of selection groups and QoS ranges can be implemented with the available resources.

To start the presentation of a discrete media object, its whole content has to be preloaded. As we assume that continuous media objects are transferred as a stream from the server to the terminal during their presentation, it is only necessary to preload the portion of the content, which is needed to bridge the delay occurring when the stream transfer is started. Since Tiempo documents are interactive, it is not possible to preload all media objects before the document presentation is started. Hence, our scheduling concept considers for each media object a separate *preloading phase* and a *presentation phase*.

Because of the preloading it is not possible to simply adopt an existing adaptive stream synchronization approach such as [8], which adjusts the playout rate to the changing resource situations. To determine instants the preloading is started at, we have to make assumptions about the resource situation in the future. Further, we have to know how much resources will be needed to preload and present objects.

7. Resource Information

To determine an optimal schedule our scheduling algorithm needs information about the amount of resources needed by the presentation and the amount of resources available while the presentation is running.

7.1 Resource Requirements of the Presentation

In the preloading phase media objects consume bandwidth and CPU cycles as well as an increasing amount of buffer needed to store the transferred data on the terminal. For our algorithm we assume that a function $V_i(v_i)$ is given which describes the amount of data that has to be transferred to start the playout of media object i with speed v_i. For discrete media objects, V_i is a constant equal to the size of the media objects content.

For continuous media objects, V_i depends on the used stream synchronization algorithm to compensate jitter and skew of media streams. In our system we apply a simple stream synchronization mechanisms that is based on globally synchronized clocks. Let us assume that we have n streams which should be synchronized at the sink (terminal) and that the maximum communication delay $d_{max,i}$ to each server i is known, with $d_{max} = max_i\{d_{max,i}\}$. A synchronized playout of the n streams is guaranteed, if the following is given. If a data-unit u_i has to be played out at the instant $T_P(u_i)$ at the terminal, it has to be sended at $T_S(u_i) = T_P(u_i) - d_{max,i} - Q_d$. Here Q_d ($Q_d \geq 0$) is a QoS parameter; how bigger it is the less delayed play-outs will occur, if the delay value is incorrect. Hence, to compensate the start delay of a stream we have to preload $V_i(v_i) := 2\ (d_{max,i} + Q_d)\ r_i\ v_i\ U_i$ data. Here r_i represents the rate of the stream i which has to be multiplied by the presentation speed of the associated media object. U_i is the average size of data-units of the stream. The maximum communication delay to a server can be gained by monitoring of the transit delays of data-units of a stream. E.g. when using RTP [9] to communicate media streams, timestamps and associated real time values of sender reports can be used to determine the occurring delay between server and terminal. To get an impression about the delay values before a stream is started, the 'ping' program, available on nearly all platforms, might be used.

We further assume that a function C(b) is given, which describes how much CPU load is produced to receive b bits per second. The function C(b) is determined by monitoring the load produced to transfer a certain amount of data from the network-adapter in the memory. Such a loading may be performed by a thread that logs how much data D is transferred per second. By monitoring the CPU load C_D produced by the thread, C(b) is given as $C(b) = C_D\ b / D$. Measurements showed that this formula is correct as long as the CPU is not overloaded.

In the presentation phase we neglect the CPU load produced to display a discrete media object since we assume this CPU load to be low. A continuous media object needs CPU resources continuously during its presentation. To determine the CPU load $C_{u,i}$ produced to playout one data-unit u of a stream i on a particular system the following technique can be applied. We select a significant subset of each continuous media object. Then we simulate a playout of the data with a rate $r_{s,i}$ specified in the document for the media object and monitor the CPU load $C_{s,i}$ produced. On the base of this load we compute $C_{u,i} = C_{s,i} / r_{s,i}$. Measurements showed that this formula is valid as long as the CPU is not overloaded and that the variance between computed and real values of $C_{u,i}$ is less than 2%. The function $C_i(v_i)$ describing how much CPU load is produced to playout a media object with speed v_i is then given as $C_i(v_i) := C_{u,i}\ r_i\ v_i$.

To be able to redisplay a discrete media object, it has to be buffered during its presentation. If a continuous media object is transferred as a stream during its presentation, buffer is needed to compensate jitter. We assume that a function $P_i(v_i)$ is given describing how much buffer is consumed to playout a media object with speed v_i. For discrete media objects P_i is equal to V_i multiplied by the compression factor. For continuous media objects P_i depends on the average size of a data unit U_i and the applied stream synchronization mechanism. The buffer needed to compensate the jitter j_i of the stream has the size $P_i(v_i) = (j_i + Q_d)\ r_i\ v_i\ U_i$. Jitter is defined as $d_{max} - d_{min}$. Hence, to get an impression about the possible jitter of a stream before it is started, we can use the d_{min} and d_{max} delivered by the 'ping' program. During the presentation the jitter of each stream is monitored.

In the presentation phase a discrete media object needs no bandwidth. Whereas if a continuous media object is transferred in real time, bandwidth is needed. The bandwidth $B_i(v_i)$ needed to present a continues media object is given as $B_i(v_i) := r_i\ v_i\ U_i$.

We further need the information which resources, such as an audio device, are needed by media objects exclusively.

7.2 Prediction of Resources Situations

Our approach is to predict the resource situation in the future based on information about the resource situation in the past.

To determine the amount of currently available buffer, the amount of free physical memory P_f on the terminal is monitored. From the computation of B_i we known how much buffer is needed for each media object. The amount of buffer currently available for the presentation is computed as $P_a = P_f + \Sigma_i B_i - Q_P$. Q_P ($Q_P \geq 0$) is a quality parameter that describes how much buffer is not assigned to the presentation.

To gain the amount of currently free CPU cycles, we observe the idle time C_f of the CPU. Further we observe the amount of CPU load $C_{p,i}$ produced by each thread dealing with the playout of continuous media items i. The amount of free CPU resources is then computed as $C_a = C_f + \Sigma_i C_{p,i} - Q_C$. Q_C ($Q_C \geq 0$) is a quality parameter. According to the value of the parameter some CPU resources are not assigned to the presentation.

To gain information about the currently available bandwidth $B_{a,s}$ to a server s we observe how much data is transferred from a server to the terminal in each second. This value is equal to the really available bandwidth $B_{r,s}$. According to the schedule we know how much data should be transferred per instant from this server $B_{v,s}$. If $B_{r,s} = B_{v,s}$ we set $B_{a,s} = B_{v,s} + d$. d is a parameter describing the amount of bandwidth that is expected to be additionally available. If $B_{r,s} < B_{v,s}$ we set $B_{a,s} = B_{r,s}$. To get an impression about the bandwidth to a server before a stream transfer or preloading, some test-packages might be transferred.

When monitoring the available resources we apply a filter of the following form

$$r_a(t) = r_a(t-1)\, g + r_a(t)\, (1-g)\ ; \qquad 0 \leq g \leq 1$$

to avoid oscillations and to get smoother modifications. Increasing g increases the influence of the new value while decreasing g results in a higher influence of the previous values.

On the base of the current values the available resources in the future are predicted. For each resource r we generate a hyperbolic function

$$R_r(t) := (\mathrm{sign}(r_a - r_m)\, (r_a - r_m)^2) / (t - t_c + | r_a - r_m |) + r_m\ .$$

In this function r_a represents the current available amount of the resource. t_c is the current instant and r_m is the average amount of resource r available in the past of the presentation. In functions constructed in this way the vertex of the hyperbolic function lies at the current instant t_c and behind t_c the function approaches towards r_m (Figure 9). The idea behind the extrapolation is the following. We know that at the moment r_a resources are available and we know that in average r_m resources were available. Hence, we assume that the amount of available resources will approach towards this value in the future.

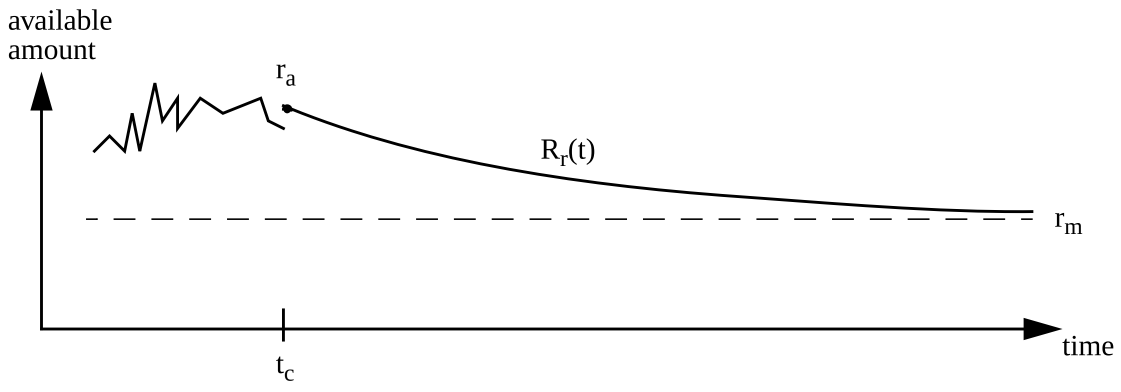

Figure 9: Prediction of available resources

8. Preparations for the Adaptive Scheduling

To generate *Mathematical Programmes*, which are used to schedule a document, we represent a specification as directed acyclic graphs called *scheduling graphs*.

8.1 Scheduling Graphs

It is not possible to efficiently solve Mathematical Programmes integrating different overlappings of media objects. Hence, we generate for each possible overlapping of media objects a separate scheduling graph.

Firstly, we generate a provisional scheduling graph for each possible combination of presentation alternatives of selection groups. A provisional scheduling graph is generated in the same way as an event-graph. For each provisional scheduling graph the possible event instants associated with the nodes are computed applying the algorithm described in Section 3. With the help of the computed event ranges it is now possible to generate scheduling graphs as follows.

We insert in a provisional scheduling graph between two nodes a and b associated with a presentation interval a directed edge from a to b and from b to a, if the event ranges of the nodes have common instants and there is no edge between a and b. Both edges are labeled with a value range [0, infinite]. Let us assume that in the example in Figure 4 the animation and the video may overlap or not overlap according to the specified QoS ranges. Then we would introduce such edges between the node representing the end of the presentation interval associated with the video and the node representing the start of the presentation interval of the animation. After that we generate for each possible combination of additional edges a separate scheduling graph. Such a scheduling graph implies totally ordered presentation interval nodes. In other words, we have a particular overlapping of media objects.

Each such scheduling graph consists of a fixed set of segments. A segment is characterized by a fix number of presented media objects. When a scheduling graph was generated, we check whether media objects which need the same resources exclusively are overlapping in a segment. If this is the case, the scheduling graph causes a resource violation and has to be omitted.

After all scheduling graphs have been generated from the provisional scheduling graphs, the scheduling graphs are arranged in a sorted list according to the object-level presentation quality they imply. The presentation quality of a scheduling graph is given by the sum of priorities of the contained presentation alternatives. All scheduling graphs derived from one provisional scheduling graph have the same object-level quality.

8.2 Generation of Mathematical Programmes

In a next step we generate for each scheduling graph in the list a Mathematical Programme. A Mathematical Programme is an optimization problem subject to constraints in $\mathfrak{R}^n$ of the form

$$\begin{aligned}&\text{Maximize } f(x)\\&\text{subject to}\\&g_i(x) = 0; \quad i = 1, \ldots, m\\&x \in S \subset \mathfrak{R}^n\end{aligned}$$

The vector $x \in \mathfrak{R}^n$ has components $x_1, \ldots, x_n$ which are the variables of the problem. The function f is called the *objective function* and the set of conditions $g_i(x) = 0$ ($i = 1, \ldots, m$) and $x \in S$ is the set of the *constraints* of the problem. The optimal solution of such a problem is a vector x^* that fulfills the constraints and maximizes the objective function. A Mathematical Programme of this form is constructed for each scheduling graph.

The scheduling graph in Figure 10 is used to illustrate the generation of a Mathematical Programme. It shows the scheduling graph of a composite media object including two continuous media objects. As in Figure 4, the grey rectangles show how edges are affected by the different speeds in the TSs.

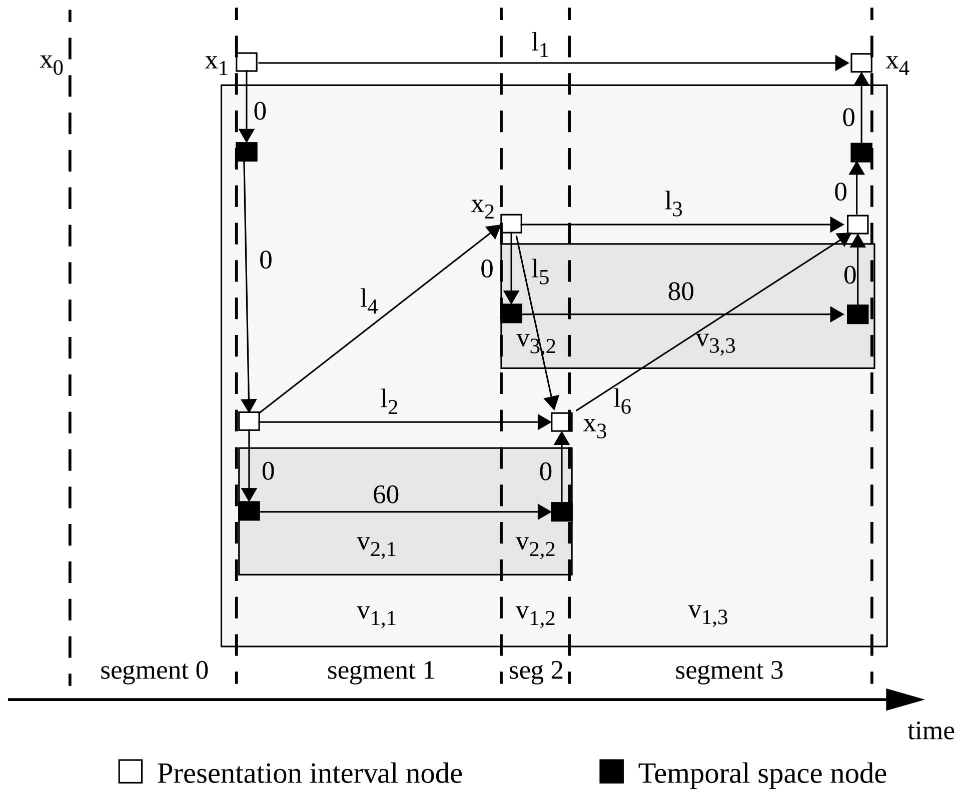

Figure 10: Scheduling graph example

Constraints representing the document specification

We introduce for each node in a scheduling graph a variable x, $x \geq 0$ representing the instant the event associated with the node occurs (If nodes are related by edges with length zero only one common variable is introduced). To be able to consider the preloading of the objects starting in the first segment, we assume that there is a further segment before the first segment given by the scheduling graph. A variable x_o represents the start of this segment. For the speed of a TS i specified by a QoS range we introduce for each segment s a variable $v_{i,s}$, $v_{i,s} \geq 0$. For flexible edges i in the graph, we introduce variables l_i, $l_i \geq 0$ representing the length of the edges. Figure 10 shows which variables have to be introduced in the scenario.

With the help of these variables a scheduling graph can be described representing each edge by a constraint. The resulting length of an edge depends on the speed in the segments the edge crosses. An edge h that crosses segment i to segment j can be represented by a constraint of the form

$$(x_{i+1} - x_i)\, v_{e,h,i} + (x_{i+2} - x_{i+1})\, v_{e,h,i+1} + \ldots + (x_j - x_{j-1})\, v_{e,h,j} = l_h$$

In this equation x_i and x_j are placeholder for the variables representing the start-instant respectively the end-instant of the edge. x_{i+1} to x_{j-1} are placeholders for variables representing the border of the segments i+1 to j-1. $v_{e,h,k}$ is a placeholder for the term $v_{a,k}\, v_{b,k}$ ...., which is the product of the speed of all TSs in segment k which contain the edge directly or indirectly. l_h is the variable that represents the length of the edge. If the edge has a constant length the right side of the equation consists of this constant.

Constraints representing resource limits

To guarantee that not more resources are required by the computed schedule than available, resource constraints are introduced. The requirements of CPU cycles, bandwidth and buffer to present the media objects o in a segment s are given as follows:

$$C_s = \Sigma_{\text{o presented in s}} C_o(v_{e,o,s})$$
$$B_s = \Sigma_{\text{o presented in s}} B_o(v_{e,o,s})$$
$$P_s = \Sigma_{\text{o presented in s}} P_o(v_{e,o,s})$$

To describe the preloading, the following constraints are introduced for each segment s where the presentation of some objects starts:

$$\Sigma_{j = 0,s\text{-}1}\, b_j\, (x_{j+1} - x_j) + w_{L,s} - m_{L,s} = \Sigma_{\text{o starts in s}} V_o(v_{e,o,s}); \quad w_{L,s} \geq 0; \quad m_{L,s} \geq 0; \quad b_j \geq 0;$$

This constraint expresses that the transfer data volume in the segments before segment s has to be greater than the preload volume of the objects started at x_s. The variable b_j represents the bandwidth used in segment j to preload objects. To express the buffer requirements for preloading we have to introduce for all segments s the following equations:

$$L_s = L_{s-1} + b_{s-1}\,(x_s - x_{s-1}) - \Sigma_{\text{o starts in s-1}} V_o(v_{e,o,s-1}); \quad L_0 = 0;$$

These equations express that the amount of needed buffer L_s at the end of a segment s is equal to the amount of buffer L_{s-1} at the start of the segment increased by the data volume transferred in the segment and decreased by the preload volume of objects whose presentation is started in the segment.

To restrict the CPU usage, we introduce for each segment s the constraint

$$C_s + C(b_s) + w_{C,s} - m_{C,s} = \min\{R_C(x_s), R_C(x_{s+1})\}\ ; \quad w_{C,s} \geq 0, \quad m_{C,s} \geq 0;$$

For the bandwidth resource we introduce for each segment s a constraint:

$$B_s + b_s + w_{B,s} - m_{B,s} = \min\{R_B(x_s), R_B(x_{s+1})\}; \quad w_{B,s} \geq 0, \quad m_{B,s} \geq 0;$$

For the buffer resource we introduce for each segment a constraint:

$$P_s + L_{s+1} + w_{P,s} - m_{P,s} = \min\{R_P(x_s), Rp(x_{s+1})\}; \quad w_{P,s} \geq 0, \quad m_{P,s} \geq 0;$$

In the equations $R_r(x_s)$ represents the predicted available amount of resource r at the segment border x_s. As it is not possible to generate constraints that reflect the detailed course of such a function, we consider the minimum value of a function $R_r(x_s)$ in each segment.

The variables $w_{r,s}$ are slack variables which describe how many units of resource r are not used in the segment. The variables $m_{r,s}$ are slack variables which describe how many units of resource r are missing in a segment. By constructing resource constraints in this way, it is also possible to compute a solution if there are not enough resources available.

Representation of QoS ranges

A QoS range of the form $[\pi_1{:}\rho_1, \pi_2{:}\rho_2, \ldots \pi_k{:}\rho_k]$ restricting a variable l or v is represented as follows:

$$l = \Sigma_{i=1,k}\, \pi_i\, l_i; \quad l_i \geq 0;$$
$$\Sigma_{i=1,k}\, l_i = 1$$
$$\Sigma_{i = 1,k;\ i\text{-}1 \bmod 2 = 0} \left[\, (\, l_i + l_{i+1})\, (\, \Sigma_{j = 1,k;\ j \neq i;\ j \neq i+1}\, l_j)\, \right] = 0$$
$$\Sigma_{i = 2,k;\ i \bmod 2 = 0} \left[\, (\, l_i + l_{i+1})\, (\, \Sigma_{j = 1,k;\ j \neq i;\ j \neq i+1}\, l_j)\, \right] = 0$$
$$W(l) = \Sigma_{i=1,k}\, \rho_i\, l_i$$

A QoS range with k anchor points is represented by three equations and k variables. The second and third equations allow only that two adjacent variables l_i are non-zero. The first equation guarantees that the value of l, which is a linear combination of the variables l_i, stays within the bounds given by the QoS range. The function W(l) assigns to each value l a priority value as defined by the QoS range. The functions W(l) are needed to construct the objective function.

Objective function

The objective function f of the Mathematical Programme is constructed as follows:

$$f = \Sigma\, W(l) + \Sigma\, W(v) - M\, \Sigma\, m_{r,j}$$

In f all functions W are added. The slack variables containing amounts of resources that are missing are inserted multiplied by a big negative number *M*. Thus, in the optimal solution (where the objective function has a maximum) these variables have minimum values and the variables representing QoS range values have values with high priorities.

Mathematical Programmes, generated as described above, contain terms where two or more different variables are multiplied. Functions containing such terms are not convex. This means, they may have more than one extremum. If such functions are contained in Mathematical Programmes, it is necessary to use rather inefficient global optimization algorithms to solve them. Whereas, there exist very efficient optimization algorithms to solve Mathematical Programmes consisting of convex functions [7]. To be able to use efficient optimization algorithms, the non-linear equations of our Mathematical Programmes can be made convex as follows.

We replace each term $x_i\, x_j \ldots x_l$ with more than two variables by a term $x_i\, k_1$ and add the equation

$$x_j \ldots x_l - k_1 = 0$$

to the constraints. This is done until all equations contain terms where less than 3 variables are multiplied. Then, each of the remaining terms $x_i\, x_j$ is replaced by a term $u_i - u_j$ $(u_i, u_j \geq 0)$ and the following equations are added to the constraints:

$$y_i^2 - u_i = 0; \quad y_j^2 - u_j = 0; \quad x_i + x_j - 2y_i = 0; \quad x_i - x_j - 2y_j = 0;$$

By inserting these equations in each other, it can be easily proven that they represent the term $x_i\, x_j$. After these transformations have been done, our Mathematical Programmes consist of linear equations and equations of the form $y_i^2 - u_i = 0$. As linear equations are convex and terms of the form $y_i^2 - u_i = 0$ are convex, we have now a convex optimization problem.

9. Adaptive Scheduling Algorithm

Whenever one of the functions R_i changes significantly, we compute an adaptation of the presentation schedule. An adaptation can be implemented not before $t_s = t_c + t_a + 2\, d_{max}$, because it takes $2\, d_{max}$ till all server can react to an adaptation and the computation of the adaptation takes also some time t_a. Hence, each adaptation is computed not for t_c but for t_s. Values t_a are estimated based on earlier adaptations.

To find an adaptation, we start with the first Mathematical Programme in the list. Let us assume that the current adaptation takes place in segment s. To represent the course of the already presented part of the specification, we omit resource constraints and constraints describing edges which are related to the presentation before s. In the remaining constraints we assign to the variable x representing the start of s the time t_s. Further, we introduce in the constraints describing the transfer volume requirements to preload objects a constant which describes the amount of already transferred data. Then we compute the optimal solution of the Mathematical Programme. If the value of the objective function is positive, we have found a solution which do not cause resource violations. If the objective function is negative, there is a resource violation. In this case we have to compute the optimal solution of the next Mathematical Programme in the list. If all Mathematical Programmes cause resource violations, we can implement the solution with the biggest objective function value or we can abort the presentation. To compute the optimal solution of a Mathematical Programme we apply the Generalized Reduced Gradient Method [7].

From the optimal solution of the 'best' Mathematical Programme start- and end-instants as well as speed values can immediately be derived for the presentation schedule. Whereas instants to start the preparation of media objects have to be determined by moving backwards through the schedule starting with the media object that will be presented last. For each media object we compute a preparation instant so that the amount of resources not used to present media objects between the preparation instant and the start instant of the object is sufficient to preload the appropriate part of the object. When the presentation schedule is updated, the playout components and server are adjusted according to the new schedule. In the RTP protocol adaptation decisions are made by the servers. In our approach adaptation decisions are made by the scheduler on the terminal. Hence, we apply the RTSP [10] protocol not only to initialize, start and stop media streams from the server to the client, but also to inform server about modified sending characteristics with the help of the Setup, Set_Parameter functionality.

10. Measurements

Table 1 shows the time needed to compute adaptations for some document specifications. The first column depicts the number of media objects in the specification. The second column shows the number of temporal relations between the media objects. The last column contains the average time that was needed to compute the schedule for a given resource situation. The specifications contained as much flexibility as possible, this means all speeds, delays and durations were described by QoS ranges. The measurements were performed on a SUN Ultra 2.

Media Objects	Temporal Relations	Time (sec)
2	2	0.04
4	5	0.1
9	16	0.6
16	28	1.7

Table 1: Scheduling Times

It can be seen that the scheduling times are acceptable. Especially if we consider that in general not all attributes in a document will be specified by QoS ranges.

The following figures show the effect of flexible specifications on the presentation quality. In all figures the left side shows the presentation with no flexibility and the right side with flexibility. The diagrams show the reciprocal of the playout rate ($1 / r_i v_i$) and the correlated playout delay for each data-unit of the presentation. Playout delays below 20 ms are omitted in the diagrams. For all measurements Q_d was set to 0.1, Q_C was set to 25, Q_P was set to 0 and d was set to 50.

Figure 11 shows the presentation of two videos. For video A we had specified a flexible playout speed between 30 frames/sec and 20 frames/sec. For the figure without adaptation we specified a playout speed of 30 frames/sec. Approximately at the half of the presentation a concurrent application was started and aborted after 30 sec. The diagram without adaptation shows that the playout delays are very high while the application is running. In the presentation with adaptation the rate of video A is reduced while the application is running. This results in much less delayed playout of frames.

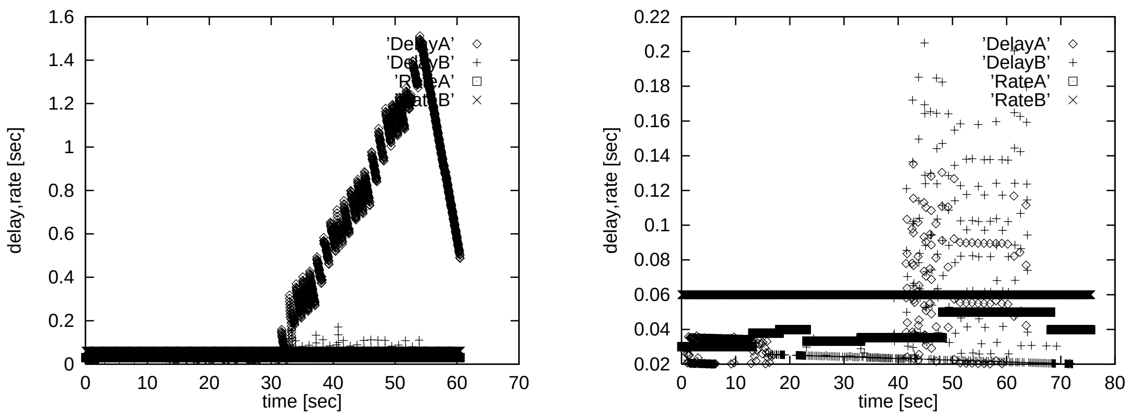

Figure 11: Playout with flexible speed

Figure 12 shows the playout with flexible speed and a selection group. Video A had a flexible speed between 30 frames/sec and 20 frames/sec and was contained in a selection group that allowed to stop playout of video A. After 20 sec we started a concurrent application running for 20 sec. The left diagram without adaptation shows that while the application is running the playout delay of both videos increases. While in the diagram with adaptation the playout of video A is stopped while the application is running. Afterwards its playout is continued.

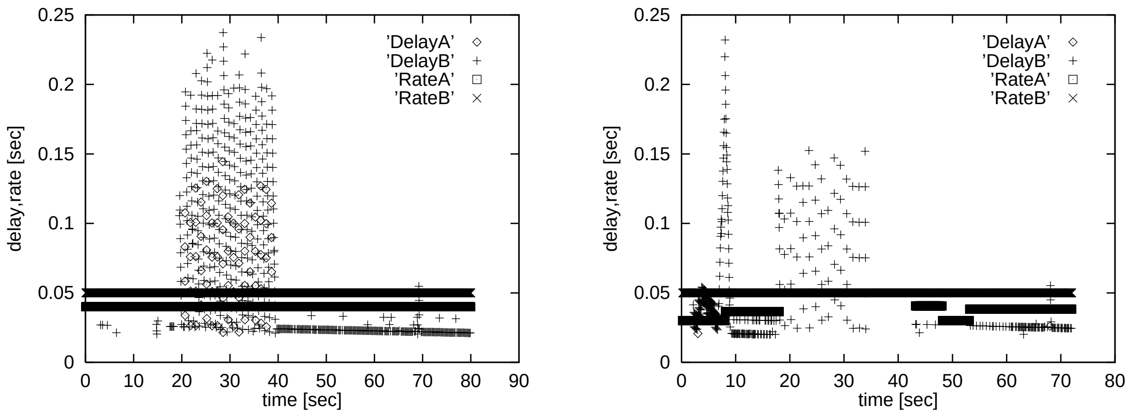

Figure 12: Playout with flexible speed and selection group

Figure 13 shows the playout of a presentation containing a selection group that allowed to playout two videos simultaneously or one after the other. If the videos are played simultaneously video B should be aborted when video A ends. If they are played one after the other, the complete videos should be played out. The diagram on the left shows what happens if the second alternative does not exist. There is an increasing playout delay for both videos. The right diagram shows what happens with selection group. The videos are played after each other and the playout delays are much lower.

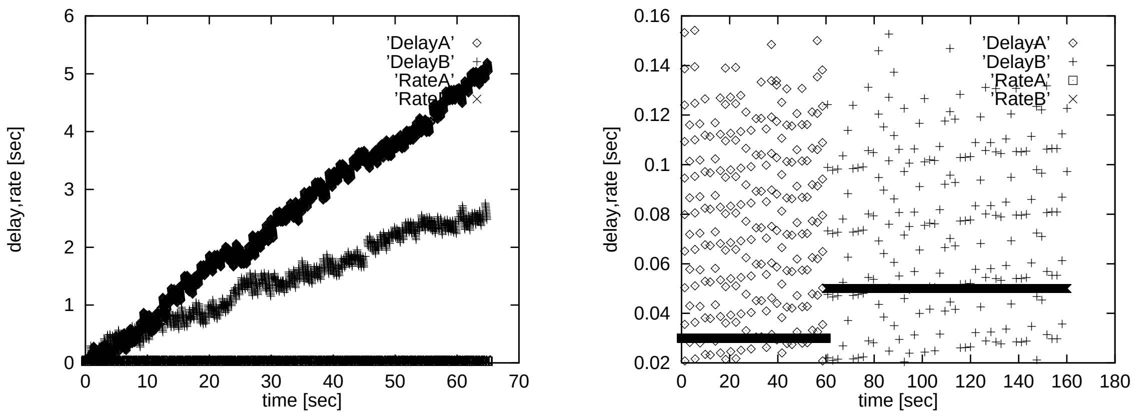

Figure 13: Playout with selection group

11. Related Work

Various temporal models and synchronization concepts have been developed for multimedia presentations, but only a few models provide for adaptive documents.

Firefly [1] is event-based, which means that the start and end of media objects are modeled as instants. The delay between related events is described by a minimum, optimal and maximum value. Additionally, costs are defined for shrinking the extent of an object to minimum or extending it to maximum. To automatically compute the optimal schedule out of these values, a linear programming algorithm has been proposed. However, the scheduling algorithm does not consider the resource situation when the presentation schedule is created.

In CHIMP [2] temporal aspects of media objects are modeled by flexible constraints, which allow to specify ranges for the temporal values. It is possible to specify alternative constraints and to assign priorities. The concept to adapt presentations to different resource situations is equivalent to the concept we propose. Available resources are also predicted and distributed between simultaneous presented media objects. Compared with CHIMP, our model allows a finer granularity with regard to the specification of priorities. Further, CHIMP does not integrate abstractions such as selection groups and it does not provide for variable presentation speeds.

Currently the W3C develops the Synchronized Multimedia Integration Language (SMIL) [5]. SMIL allows integrating a set of independent multimedia objects into a synchronized presentation. Using SMIL, an author can describe the temporal behavior of the presentation, describe the layout of the presentation on the screen as well as associate hyperlinks with media objects. SMIL offers a construct called ‘switch’ that allows to specify alternative media objects respectively presentations. The order of the alternatives in a switch construct defines their priorities. A SMIL interpreter selects the first alternative in the switch-construct that fulfills so-called test-attributes. These test-attributes allow the author to define requirements that have to be fulfilled to present the associated alternative. With regard to QoS aspects there exists only the test-attribute ‘system-bitrate’ defining how much bandwidth is needed to present the alternative. In contrast to our approach, SMIL documents are not continuously adapted to changing resource situations. Adaptations in form of selections of switch-statement alternatives are only performed at the start of these statements. SMIL does not provide attribute-level adaptivity.

12. Conclusion

To perform multimedia presentations under different resource situations, adaptable document models are required. The Tiempo model offers selection groups to represent alternative presentation parts consisting of media objects and interval operators. With QoS ranges alternative presentation behavior of media objects or interval operators can be defined. Assigned priorities define which alternatives of selection groups or QoS ranges should be preferred. Hence, Tiempo conform multimedia documents can integrate a high degree of flexibility and resource shortages need not result in a reduced presentation quality.

The developed consistency checking algorithm permits to detect errors in flexible specifications. It is applied in the Tiempo document editor which allows a mainly graphical composition of multimedia documents. The editor visualizes documents in different views and integrates object-oriented interaction concepts.

The adaptive scheduling algorithm is designed for environments with best-effort assignment of resources. It allows to adapt flexible presentations to changing resource situations. Whenever an adaptation is necessary, the algorithm selects presentation alternatives such that the available resources are distributed optimal between simultaneously presented media objects. Measurements showed that the algorithm can help to improve the presentation quality.

13. References

[1] M. Cecelia Buchanan and Polle T. Zellweger. “Scheduling Multimedia Documents Using Temporal Constraints”, in Proc. International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, USA, pp. 223–235, 1992.

[2] K. Selcuk Candan, B. Prabhakaran and V.S. Subrahmania. “CHIMP: A Framework for Supporting Distributed Multimedia Document Authoring and Presentation”, in Proc. ACM Intl. Conference on Multimedia, Boston, USA, pp. 329-339, 1996.

[3] Dechter, Meiri, Pearl. Temporal Constraint Networks. In Proc. *First International Conference on Principles of Knowledge Representation and Reasoning*, 5 1989, S. 83-93.

[4] HyTime. “Information technology - Hypermedia/Time-based Structuring Language (HyTime)”. ISO/IEC 10744, 8 1992.

[5] Philipp Hoschka (Ed.). “Synchronized Multimedia Integration Language (SMIL) 1.0 Specification”, W3C Proposed Recommendation, April 1998.

[6] MHEG. “Information Technology - Coding of Multimedia and Hypermedia Information”, ISO/IEC 13522, 1994.

[7] M. Minoux. “Mathematical Programming - Theory and Algorithms”, John Wiley and Sons. 1986.

[8] Kurt Rothermel, Tobias Helbig. “An Adaptive Protocol for Synchronizing Media Streams”, In ACM/Springer Multimedia Systems, Vol. 5 No. 5, September 1997, pp. 324-336

[9] Henning Schulzrinne. “Internet Services: from Electronic Mail to Real-Time Multimedia”. In Tagungsband GI/ITG Kommunikation in Verteilten Systemen, FRG Chemnitz, pp. 21-34, 2 1995.

[10] H. Schulzrinne, A. Rao, R. Lanphier. “Real Time Streaming Protocol (RTSP)”. Internet-Draft, MMusic WG, 2 1998.

[11] Robert Sedgewick. “Algorithms”, Second Edition, Addison-Wesley, 1999, S. 476-478.

[12] Thomas Wahl and Kurt Rothermel. “Representing Time in Multimedia Systems,” in Proc. IEEE Intl. Conference on Multimedia Computing and Systems, Boston, USA, pp. 538–543, 1994.

[13] Thomas Wahl, Stefan Wirag and Kurt Rothermel. “TIEMPO: Temporal Modeling and Authoring of Interactive Multimedia”, In Proc. IEEE Intl. Conference on Multimedia Computing and Systems, Washington DC, 5 1995, pp. 274-277.

[14] Stefan Wirag. “Adaptive Scheduling of Multimedia Documents”. Fakultätsbericht 1997/12, Universität Stuttgart, 7 1997.

[15] Stefan Wirag. “Modeling of Adaptable Multimedia Documents”, In Proc. Interactive Distributed Multimedia Systems and Telecommunication Services; International Workshop, IDMS’97, Darmstadt, Germany, September 1997, pp. 420-429.

[16] Stefan Wirag, Kurt Rothermel. “Adaptive Multimedia Documents”. Journal of Computing and Information Technology (CIT). Vol. 6, No. 3, September 1998.