Modellierung und Ablaufplanung adaptiver Multimedia-Dokumente

Von der Fakultät Informatik der Universität Stuttgart
zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Stefan Wirag

aus Sindelfingen

Hauptberichter:	Prof. Dr. Dr. K. Rothermel
Mitberichter:	Prof. Dr. W. Effelsberg
Tag der mündlichen Prüfung:	24. Mai 2002

Institut für Parallele und Verteilte Systeme der Universität Stuttgart
2003

Vorwort

Diese Abhandlung entstand während meiner Arbeit als wissenschaftlicher Angestellter am Institut für Parallele und Verteilte Systeme der Universität Stuttgart. Die Konzepte der Arbeit wurden im Rahmen des von der Deutschen Forschungsgemeinschaft (DFG) geförderten Tiempo-Projekts erarbeitet und validiert.

An erster Stelle möchte ich dem Hauptberichter, Herrn Prof. Dr. Dr. Kurt Rothermel, für die wertvollen Diskussionen und Hinweise zum Thema, sowie die Betreuung der Arbeit danken. Bei Herrn Prof. Dr. Wolfgang Effelsberg bedanke ich mich für die Übernahme des Mitberichts und seine Empfehlungen zur Arbeit.

Der Deutschen Forschungsgemeinschaft (DFG) danke ich für die Finanzierung meiner Forschungen im Rahmen des Tiempo-Projekts.

Für die Diskussionen zum Thema und die Zuammenarbeit in der ersten Phase des Tiempo-Projekts bedanke ich mich bei Herrn Thomas Wahl. Die kritische Hinterfragung des Themas und die damit verbundenen Diskussionen mit meinen Kollegen Jürgen Hauser, Jürgen Klarmann, Frank Sembach, Dr. Gabriel Dermler und Dr. Ingo Barth haben die Ideen der Arbeit verfestigt. Mit ihren Studien- und Diplomarbeiten haben die Herren Steffen Fischer, Georg Franke und Markus Mayer zur Implementierung und damit zur Validierung und Verfeinerung der Konzepte beigetragen. Dafür möchte ich mich an dieser Stelle bedanken.

Sindelfingen, im Mai 2002

Inhaltsverzeichnis

Abkürzungsverzeichnis

ATM	Asynchronous Transfer Mode
CD-ROM	Compact-Disc-Read-Only-Memory
CINEMA	Configurable Integrated Multimedia Architecture
CPU	Central Processing Unit
DVD	Digital Versatile Disk
DTD	Document Type Definition
FSP	File Selection Panel
GIF	Kompressionsstandard
H.261	Kompressionsstandard
H.263	Kompressionsstandard
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
HyTime	Hypermedia/Time-based Structuring Language
IP	Internet Protocol
JPEG	Joint Picture Expert Group
KByte	Kilobyte
MAVA	Multimedia On-Demand Versatile Architecture
MHEG	Multimedia/Hypermedia Information Coding Expert Group
MPEG	Motion Picture Expert Group
MP3	Kompressionsstandard
NFS	Network File System
PC	Personal Computer
PCO	Presentation Control Object

PDA	Persönlicher Digitaler Assistent
QoS	Quality of Service
RSVP	Resource Reservation Protocol
RTP	Real Time Protocol
RTCP	Real Time Control Protocol
RTSP	Real Time Streaming Protocol
TCP	Transmission Control Protocol
Tiempo	Temporal Integrated Model to Present Multimedia Objects
TSP	Type Selection Panel
UDP	Universal Datagram Protocol
WAV	Kompressionsstandard
WWW	World Wide Web
WYSIWYG	What You See Is What You Get
XML	Extensible Markup Language
XPM	Kompressionsstandard

Zusammenfassung

Multimedia-Dokumente ermöglichen eine koordinierte integrierte Präsentation diskreter und kontinuierlicher Medien. Um eine hohe Verfügbarkeit zu erreichen, werden Multimedia-Dokumente auf Servern gespeichert. Von dort können sie bei Bedarf über ein Netzwerk abgerufen und auf einem Terminal präsentiert werden. Durch die enthaltenen kontinuierlichen Medien erfordert die Präsentation von Multimedia-Dokumenten viele Ressourcen, so daß schnell Ressourcenengpässe entstehen, wodurch die Präsentationsqualität negativ beeinflußt wird bzw. eine Präsentation unmöglich werden kann. Eine Lösung für dieses Problem bieten adaptive Multimedia-Dokumente, die sich an unterschiedliche Ressourcensituationen anpassen können und somit auf unterschiedlichen Terminals und in verschiedenen Systemumgebungen präsentiert werden können.

In dieser Arbeit wird ein Modell für adaptive Multimedia-Dokumente vorgestellt und es werden Implikationen erörtert, die sich durch die Adaptivität in Bezug auf die Dokumenterstellung, die Konsistenzwahrung, die Ablaufplanung und die Architektur eines Dokumentensystems ergeben.

Adaptive Multimedia-Dokumente enthalten alternative Präsentationsmöglichkeiten, die je nach Ressourcensituation ausgewählt werden. Das vorgestellte Dokumentenmodell integriert unterschiedliche Abstraktionen zur Definition von alternativen Präsentationsmöglichkeiten sowie zur Steuerung der Auswahl geeigneter Alternativen. Der Schwerpunkt liegt hierbei auf der Modellierung adaptiver zeitlicher Aspekte.

Die Spezifikation von Präsentationsalternativen führt zu einer komplexeren Dokumentstruktur. Es wird daher ein graphischer Dokumenteditor erforderlich, mit Hilfe dessen adaptive Dokumente übersichtlich erstellt werden können. In dieser Arbeit wird ein Vorschlag für die Benutzerschnittstelle eines solchen Editors erörtert. Die komplexere Dokumentstruktur hat zudem eine höhere Fehleranfälligkeit zur Folge, da insbesondere zeitliche Abhängigkeiten zwischen Medien nicht mehr so einfach zu überblicken sind. Des weiteren ist eine manuelle Fehlersuche mühsam und zeitintensiv. Zur Abhilfe wird ein automatischer Konsistenzprüfungsmechanismus vorgestellt.

Zur Präsentation adaptiver Dokumente ist ein Ablaufplanungsmechanismus erforderlich, der in Abhängigkeit der vorherrschenden Ressourcensituation einer Präsentationsumgebung die bestmögliche Präsentation erzeugt, indem er geeignete Präsentationsalternativen auswählt. Der vorgestellte Ablaufplanungsalgorithmus für das beschriebene adaptive Dokumentenmodell, kann sowohl in Systemumgebungen in denen Garantien für Ressourcen existieren, als auch in Umgebungen ohne solche Garantien eingesetzt werden.

Die in der Arbeit vorgestellten Konzepte wurden prototypisch als Multimedia-Dokumentensystem implementiert, das über standardisierte Internet-Protokolle mit Dokument-Servern kommuniziert. Da Multimedia-Dokumente einem ständigen Wandel unterworfen sind, was die Gestaltung und die verwendeten Medienarten betrifft, sollten Dokumentensysteme einfach um neue Medienarten und Darstellungselemente erweiterbar sein. Das entwickelte Dokumentensystem basiert daher auf einer Architektur, die Erweiterungen explizit unterstützt.

Abstract

Multimedia documents are of importance in several application areas, such as education, advertising, and entertainment. Multimedia documents present discrete media objects, such as graphic or text, and continuous media objects, such as video, audio, or animation, in an orchestrated manner to the user. In a distributed environment, multimedia 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 PC, a Set-Top-Unit, or even a mobile device. When a user initiates the presentation of a 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 data units etc.

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. Since multimedia documents comprise continuous media, the presentation of those documents may require a significant amount of resources. The amount of resources available strongly depends on the system configuration and the current system load. Clearly, a workstation connected to a high-speed network allows for a higher quality presentation than a Personal Digital Assistant (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. A comparison of the approaches shows that the second approach has advantages, since it avoids redundancy and does not have to predict numerous configurations.

Even if we 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. If the underlying system provides for resource reservation, the resources needed to present a (monomedia) 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 multimedia 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 released incrementally while the presentation progresses, which again can lead to resource shortages.

When a resource shortage occurs there are basically three ways to react to it. First, just ignore it and accept the quality of the presentation to be decreased somehow. Second, abort the presentation, and third, adapt the presentation in a controlled manner to the given resource situation. To enable the last alternative which is the best solution for the user, adaptive multimedia document models and adaptive scheduling algorithms are required that allow to compile presentations for different resource situation.

This thesis presents a model for adaptive multimedia documents and discusses the effect of adaptability on document specification, document consistency, scheduling of presentations, and multimedia document system architecture. The thesis is focusing on adaptability in the temporal dimension.

Adaptive multimedia documents contain presentation alternatives implying different resource requirements. When a multimedia document is prepared for presentation, an appropriate sub-set of presentation alternatives is selected taking account of the actual resource situation at presentation time. Presentation alternatives can be specified on different levels. Alternative media objects, alternative temporal relations of media objects, optional media objects, or optional temporal relations between media objects can be defined at the level of document elements. Flexible presentation durations, flexible presentation rates for media objects, or flexible temporal delays between media objects can be defined at the level of document element attributes. Further on, it must be possible to control the selection of presentation alternatives by selection policies or quality metrics. Document models that integrate such concepts are called adaptive multimedia document models.

There already exist document models that enable to specify simple adaptive multimedia documents [WiRo98]. The models Firefly [BuZe92, BuZe93] and Mbuild [HSR92, HSR94] offer abstractions to specify adaptability at attribute level. Additionally, the models CHIMP [CPS96] and MODE [BHL91] provide simple abstractions for adaptability at document element level.

The existing document models do not support all relevant variations of presentation alternatives and the means provided to control the selection of presentation alternatives are not sufficient. The multimedia document model Tiempo (Temporal Integrated Model to Present Multimedia Objects) [Wira97], which was developed at the Institute of Parallel and Distributed Systems of the University of Stuttgart, presents possible solutions for the problem areas. In the Tiempo-model, temporal and spatial aspects of media objects are represented by so called data spaces. A data space is described by finite coordinate axes, whose length is equal to the extent of the media object in the corresponding dimension. The data units of media objects are arranged within this coordinate system. Simple data spaces represent simple media objects, such as videos, audios, pictures, texts etc. Composite data spaces represent multimedia objects, such as a page or scene of a multimedia document, and comprise arranged simple and composite data spaces. Data spaces are spatially arranged within other data spaces by absolute coordinates. Ten so called interval operators [WaRo94] can be used to arrange data spaces temporally. The interval operators are intuitive and enable to specify all relevant temporal relations. Projections allow to change the properties of data spaces, e.g. only a particular part of a data space can be presented, parts of data spaces can be combined to create new virtual media objects, or the presentation speed can be changed.

Some multimedia presentations, such as multimedia lectures, should offer users interaction possibilities, for example to repeat sections of the rendered lecture. Hence, the Tiempo-model integrates a complete set of temporal interactions [WRW95, WWR95]. An interaction possibility is represented by an interaction manager that defines via a reaction operator interaction effects on data spaces, projections, or interval operators. Interaction effects are specified by action elements. Tiempo supports elementary actions to start and stop media objects, to change the presentation speed of media objects, or to jump within a presentation. Composite actions can be composed to describe complex interaction effects.

The Tiempo-model comprises a set of abstractions to specify advanced adaptability aspects [Wira97, WiRo98]. Tiempo enables the specification of adaptability at document element level by so called selection groups and at attribute level by so called Quality-of-Service (QoS) ranges. A selection group defines a set of presentation alternatives. Each presentation alternative can represents an arbitrary part of a presentation which may consist of several media objects, interval operators, or selection groups. At presentation time exactly one presentation alternative of

each selection group is rendered at the same time. Since selection groups can be nested, the model allows to specify different levels of adaptation. Priorities can be assigned to presentation alternatives to define the presentation quality that is associated with them. The model provides three selection policies that can be assigned to a selection group to define when a presentation alternative can be selected or when it is allowed to switch from one presentation alternative to another. Hence, it is possible to specify the sequence in which presentation alternatives will be chosen when resource shortages occur. A QoS range defines a value interval where a priority can be assigned to each value. The priority defines the presentation quality of the value. Further on, a selection policy can be assigned to a QoS range to define when it is allowed to select a value or switch from one value to another. QoS ranges can be used to specify presentation rates and presentation durations of media objects as well as delays implied by interval operators. Especially the combination of QoS ranges and projections allows to specify fine granular adaptation possibilities. For example, it can be specified that independent of the actual length of a video, a particular scene shall always be presented in the middle of the multimedia presentation, which is impossible in other adaptive document models.

The specification of presentation alternatives increases the complexity of the document structure. Further on, multimedia presentations are often composed by marketing people, lecturers or teachers who have only limited programming knowledge. Hence, software tools called authoring systems are needed which allow to compose multimedia presentations by visual programming techniques.

The creation of an interactive multimedia presentation via an authoring system can be decomposed into several steps. Within a media generation phase, elementary media objects are created, for example by recording a video-clip or a piece of music. Then, in a composition phase, those media objects are assembled to a multimedia document. This phase comprises three subphases. Temporal relations and interactions between media objects are specified within a structuring phase. The position of media objects on the terminal screen is defined within a layout phase. In a test phase, the author validates whether the composed presentation accomplishes the requirements. Those phases can be repeated several times until the author is satisfied with the produced multimedia presentation. To create an adaptive multimedia document, the author has to specify presentation alternatives in the structuring and layout phase. This implies new requirements for authoring systems.

An authoring system should allow the individual selection, visualization and manipulation of each valid combination of presentation alternatives while composing a document. It should be possible to test the selection of presentation alternatives like it will happen during a presentation. An authoring system should provide information about possible values of adaptive attributes to help authors to deal with the higher complexity of the document structure and to avoid errors in advance.

This thesis presents a concept for an authoring system for adaptive multimedia documents [Wira99a]. The concept comprises three different views to visualize and edit different aspects of a multimedia document.

Each document element that may contain other document elements has a main view. The main view is based on a flowchart visualization technique, where document elements such as media objects, interval operators, selection groups etc. are represented by appropriate icons. Those icons are connected by edges to show the relations of the document elements. Document elements are inserted in the main view via drag&drop. In this way the (hierarchical) structure of a multimedia document is composed. Each document element has a menu to set properties such as the attribute-level adaptability.

The main view integrates an automatic completion of the hierarchical document structure. If an author tries to insert a document element that can not be contained directly in the surrounding document element, the authoring system inserts automatically missing intermediate elements. To prevent temporal specification errors, the authoring system integrates a consistency checking mechanism that is invoked if a document element is inserted or deleted as well as if the attribute of an element is changed. Additionally, the consistency checking mechanism determines the maximum QoS ranges which are used as default attribute values. Hence, an author is always informed about possible attribute values and manual computations are superfluous.

The temporal layout of a document is visualized in a temporal view, which is based on a time-line visualization technique. The presentation of media objects implied by projections is visualized by bars on time-lines. The bars are arranged like defined by interval operators. The temporal view shows the active presentation alternative of each selection group, which can be selected in the main view. The temporal view provides sliders to simulate different resource sit-

uations and shows the correlated temporal layout of the optimal presentation for this amount of resources. Hence, different combinations of presentation alternatives can be simulated, QoS ranges can be determined in detail, or the effectiveness of selection groups can be validated.

Each document element that is spatially enclosed has a spatial view. Such document elements are associated with the terminal screen or a window on that screen. In a spatial view, media objects are arranged and displayed like in the presentation (WYSIWYG approach). As different media objects can be located at the same position during a presentation, the instant shown in the spatial view can be selected in the temporal view. A spatial view allows to define and manipulate the position, size, and overlapping of media objects.

A multimedia presentation can only be rendered if the multimedia document contains no errors. In other words, it has to be consistent. The consistency on document element level is given if document elements have the required attributes and if their hierarchical structure is correct. Consistency on attribute level requires that specified attribute values are conform to the attribute type. Further on, certain relationships between temporal attributes of media objects and interval operators have to be preserved.

To verify its consistency, authors can pass a document specification to a presentation system, and fix indicated errors step-by-step, until the document specification is faultless. Such a manual consistency checking is suggested by most existing document systems. Anyhow, a manual testing approach has disadvantages. If documents are complex and contain many interaction possibilities, it is almost impossible to manually test all paths through the presentation in a reasonable amount of time. Hence, even professional multimedia documents often contain errors, e.g. buttons that have no effect. Considering the presentation alternatives of adaptive documents, manual testing is no longer feasible at all, because it is impracticable to generate for each presentation alternative combination the correlated resource situation to validate it. A possible solution for this problem is an automatic consistency checking of document specifications by a tool called consistency checker that simulates all possible presentation situations to find contained errors.

This thesis presents an algorithm based on directed acyclic graphs, so called event graphs, to check the temporal consistency of document specifications. The nodes of an event graph repre-

sent events such as the start or end of a media object presentation. The edges describe the temporal delays between the events. Edges have a minimal and maximal length if the represented delay was specified by a QoS range. A specification conform to the Tiempo model can be completely represented by an event graph. A separate event graph is build for each possible combination of presentation alternatives of selection groups. In order to validate the temporal consistency, it has to be checked whether there are contradicting delays between the nodes of an event graph. That can be checked applying the algorithm of Dechter, Meiri and Pearl [DMP89], which is based on the Floyd-Warshall algorithm. Since this algorithm implies a cubic complexity, the following optimizations are proposed. If data spaces are nested, each data space in the hierarchy is validated separately, and not the entire hierarchy at once. Hence, the Floyd-Warshall algorithm will be applied to smaller event graphs. If documents contain presentation alternatives, data space lengths computed for a combination of presentation alternatives are buffered and reused. Thus, repeated validation of already considered data spaces is avoided.

Before a multimedia document can be rendered, the presentation system has to compile a schedule that describes when which actions have to be performed to implement a multimedia presentation according to the multimedia document. The process to generate a schedule is called scheduling. The objective of adaptive scheduling is to avoid resource shortages by considering the resource situation when creating schedules [Wira99b, Wira99c].

The presentation process for a media object comprises two main phases. In a preparation phase, media object data is transferred from the server to the terminal, so that the presentation phase, where the media object content is presented to the user, can start at a pre-determined instant. In each of those phases one or more presentation actions, which consume system resources, have to be processed. The total resource requirements of a multimedia presentation at an instant is equivalent to the accumulated resource requirements of the simultaneously processed presentation actions. With regard to the instant when a resource requirement has to be fulfilled, position-dependent and position-independent resource requirements can be distinguished. Position-dependent resource requirements have to be satisfied at a given instant, otherwise the presentation quality would decrease immediately. To this category belong the storage, the CPU-cycles and bandwidth required in the presentation phase of continuous media objects. Position-independent resource requirements can be postponed within given limits. This means, it is not mandatory that a certain amount of resources is available at a given instant, but that the required

amount of resources is vacant within a pre-determined interval. The CPU-cycles and bandwidth required in the preparation phase of media objects belong to this category.

The aim of adaptive scheduling algorithms is to schedule presentation actions in such a way that enough resources are available to complete them on time. Adaptive scheduling algorithms can be classified with regard to the interval when adaptations are computed and when the preparation phases are processed. If only one adaptation is computed before the presentation starts, they are called static scheduling algorithms. If adaptations are computed during the whole presentation interval, they are called dynamic scheduling algorithms. Dynamic algorithms can be subclassified in proactive and reactive scheduling algorithms. Reactive algorithms do not exploit information about needed or available resources. Hence, several steps are necessary to determine the optimal adaptation by trial and error. In contrast, proactive scheduling algorithms employ resource information to compute adaptations. Those algorithms compare required and available resources, and if a resource shortage is detected, an adaptation is computed. The Tiempo scheduling algorithm is a dynamic proactive scheduling algorithm. Thus, it needs information about required and available resources.

The properties of media objects represent the basis to determine resource requirements. It can be differentiated between system-dependent and system-independent resource requirements. System-independent resource requirements are determined solely from the properties of a media object, such as the bandwidth needed to transfer the media data and the storage to buffer those data. System-dependent resource requirements, such as the CPU-cycles needed to decode and render media data, are defined by the properties of a media object, the properties of the presentation terminal, as well as the properties of presentation components and decoding algorithms. System-dependent resource requirements have to be ascertained by measurements, for example with the help of a testbed that is embedded in the presentation system.

When adaptations are computed, it is necessary to consider the entire remainder of a presentation to prevent potential resource shortages in the future. To enable the prediction of available resources in the future, information about available resources are collected periodically during the whole document presentation. The Tiempo approach predicts the available storage, CPU-cycles, and bandwidth by assuming that in the future the same amount of resources will be available as in the recent past.

Considering those resource information and the specified priorities for presentation alternatives, the Tiempo scheduling algorithm computes the optimal schedule. It is invoked whenever the mismatch between the resource requirements of the presentation and the available resources becomes bigger than a threshold.

Event graphs are used to represent the different presentation alternative combinations during the scheduling. Since scheduling tests for resource requirements of presentation actions can only be performed efficiently if the overlapping of media object presentation phases is unique, a multimedia document is transformed into unique event graphs before the scheduling starts. Each of those event graph implies a certain presentation quality that is given by the sum of priorities of the incorporated presentation alternatives on document element level.

The Tiempo scheduling algorithm includes a presentation action scheduling algorithm which allows to compute a schedule for an unique event graph that implies minimal resource consumption and does not violate temporal constraints of the document specification. When computing such a schedule, the algorithm tries in a first step to schedule presentation actions with position-dependent resource requirements. Then, it attempts to schedule presentation actions with position-independent resource requirements. Presentation actions with position-independent resource requirements are processed after those with position-dependent resource requirements, since they can be moved within certain limits. If no resource conflicts are detected in these steps, a resource optimal schedule for the event graph has been found.

Generally, a resource optimal schedule is not optimal with regard to the specified QoS range priorities. Therefore, the Tiempo scheduling algorithm includes an optimization algorithm that allows to compute an optimal event graph schedule. Such a schedule is optimal with regard to the priorities on attribute level and it causes no resource shortage. The thesis proposes two approaches to consider priorities of QoS ranges during the scheduling. In the first approach, the sum of priorities of chosen QoS range values is maximized. In the second approach, the priorities direct the sequence how the flexibility of attributes is eliminated. The first approach is NP-complete, while the second approach has a polynomial complexity. Hence, the first approach can only be used with small presentations, because for large presentations the scheduling will take too long. With both approaches, the optimal event graph schedule is computed iteratively.

In each optimization step, the presentation action scheduling algorithm is invoked to test if it is still possible to schedule all presentation actions of the event graph.

In order to find the overall optimal schedule, the Tiempo scheduling algorithm processes the unique event graphs in sequence of descending presentation quality. The scheduling algorithm starts with the event graph that implies the best presentation quality. Applying the optimization algorithm, it is tried to compute a schedule. If this is possible without resource conflicts, the presentation will be controlled according to that schedule, until the next adaptation occurs. If a resource conflict is detected, the next best event graph will be processed and so on, until a graph is found that does not causes resource conflicts. If all event graphs imply resource conflicts and the presentation shall not be terminated, the event graph that implies the smallest resource shortage is chosen as optimal schedule.

The Tiempo scheduling algorithm can be applied in combination with resource reservation or with best-effort resource assignment. Only the way how the actions of a schedule are implemented at the underlying system is different.

The presented concepts have been implemented in a multimedia document system prototype that comprises an authoring system and a presentation system [Wira99a]. The Tiempo document system communicates via the standardized internet protocols RTP, RTSP, and HTTP with document server to access multimedia documents and media objects.

Considering the design of multimedia documents and the contained media object types, user demands grow continuously. Hence, document systems should be easily extendable with respect to new media types and the presentation of media types. The Tiempo document system is based on an architecture that explicitly supports extensions. In this architecture, each document element of a multimedia document is represented by a so called presentation control object. A presentation control object integrates the complete editing and rendering functionality that is needed to process the document element within the authoring and presentation system. All presentation control objects are derived from each other. Applying this object oriented approach, it is easy to extend the document system by new media types or presentation concepts, since new presentation control objects can be integrated via dynamic load libraries in the multimedia document system.

1 Einleitung

1.1 Motivation und Ansatz

Dokumente, wie Bücher, Zeitschriften, Zeitungen oder Lexika, dienen seit jeher zur Wissensvermittlung, Unterhaltung oder Werbung. Durch die zunehmende Verbreitung von Computern werden Dokumente inzwischen nicht nur digital erstellt, sondern in Form digitaler Dokumente auch auf elektronischem Wege verteilt und betrachtet. Zur Zeit werden zahlreiche sogenannte Digital-Library-Projekte durchgeführt, die das Ziel haben, eine Infrastruktur für digitale Bibliotheken zu entwickeln, die einem breiten Publikum den online-Zugriff auf digitale Dokumente ermöglicht.

Die ersten digitalen Dokumentensysteme, die einen Mehrwert durch den Einsatz von Computern boten, waren Hypertext-Systeme, wie beispielsweise HyperCard [Good90]. Ein Hypertext-System besteht aus einer Menge von Dokumenten, die durch Verweise (auch Hyperlinks genannt) vernetzt sind. Die Informationen in einem Hypertext-System haben somit nicht, wie bei einem gedruckten Dokument, eine sequentielle Struktur, sondern bilden ein Netzwerk-Struktur. Dies hat den Vorteil, daß Abhängigkeiten zwischen Informationseinheiten hervorragend modelliert werden können und somit auch komplexe Sachverhalte gut zu vermitteln sind. Bei der Präsentation eines Hypertext-Dokuments werden Verweise visualisiert. Der Nutzer kann dann, meist durch Anklicken, dem Verweis folgen und das Dokument, zu dem der Verweis führt, abrufen. Diesen Vorgang nennt man Navigation in einem Hypertext. Auf diese Weise hat der Nutzer die Möglichkeit, Informationen in der von ihm gewünschten Reihenfolge abzurufen und zu konsumieren. Dokumente in Hypertext-Systemen beinhalten nur wenig nicht-textuelle Informationen, wie z.B. Bilder. Da diese Informationen keinen inhärenten Zeitbezug besitzen, nennt man sie zeitunabhängige oder diskrete Medien.

Durch die gestiegene Leistungsfähigkeit von Computern und die Integration entsprechender Verarbeitungskomponenten [SRR90] enthalten digitale Dokumente inzwischen auch Medien mit inhärentem Zeitbezug, wie Videos, Audios und Animationen. Der Einsatz solcher kontinuierlicher Medien ermöglicht eine verbesserte Wissensvermittlung, da einerseits neue Inhalte vermittelt werden können, wie beispielsweise ein Musikstück, und andererseits Sachverhalte

anschaulich dargestellt werden können, wie beispielsweise ein physikalischer Vorgang mittels eines Videos oder einer Animation.

Setzt sich ein Dokument aus diskreten und kontinuierlichen Medien zusammen, nennt man es Multimedia-Dokument [Stei93b, BDF+92]. Werden mehrere Multimedia-Dokumente durch Verweise verknüpft, entsteht ein Hypermedia-Dokument. Abbildung 1-1 zeigt die schematische Darstellung eines Hypermedia-Dokuments. Dieses Beispiel zeigt vier Teildokumente, die über Verweise miteinander verknüpft sind. Das Teildokument links unten integriert kontinuierliche Medien, und zwar ein Video, das ein rotierendes Auto zeigt, und eine Hintergrundmusik, die während des Videos abgespielt wird.

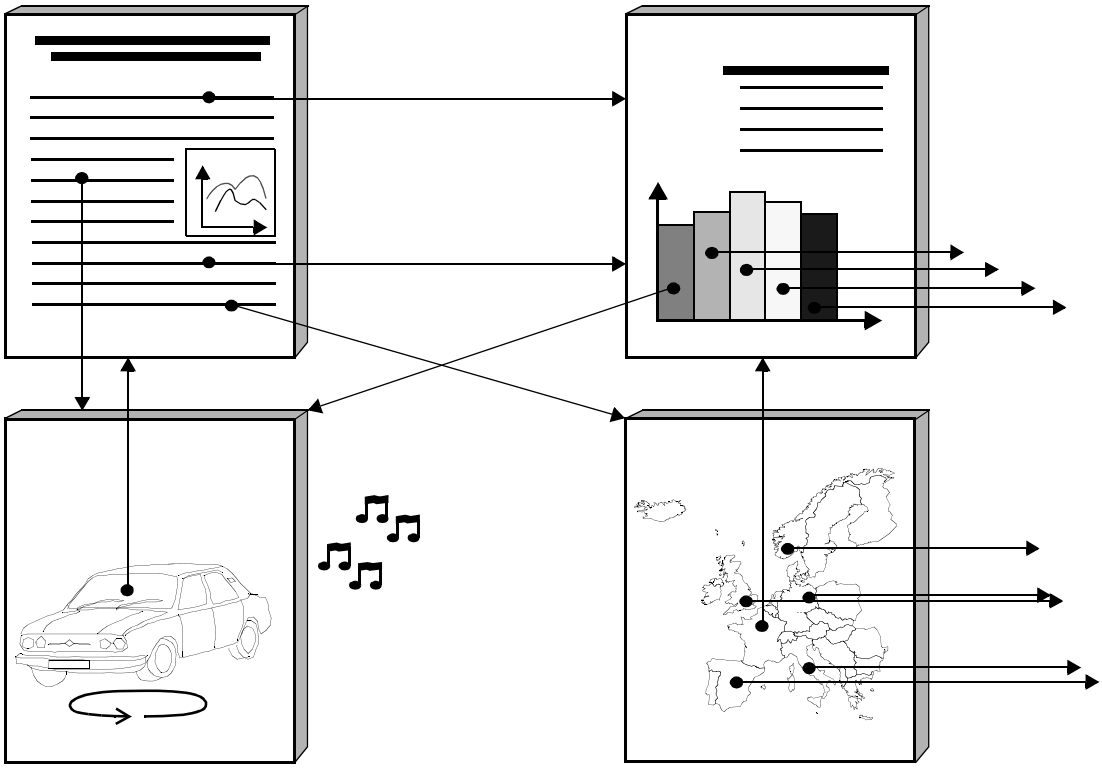


Abb. 1-1 : Hypermedia-Dokument

Ehe ein Hypermedia-Dokument betrachtet werden kann, müssen die Multimedia-Dokumente und Verweise spezifiziert werden aus denen es besteht. Damit ein aus mehreren Medien bestehendes Multimedia-Dokument ansprechend präsentiert werden kann, muß definiert sein, wie die enthaltenen Medien räumlich auf dem Bildschirm des Nutzers plaziert werden sollen.

Da kontinuierliche Medien eine zeitliche Dimension besitzen, muß des weiteren festgelegt werden, in welcher zeitlichen Abfolge die Medien dem Nutzer präsentiert werden sollen. In diesem Zusammenhang spricht man auch von der Synchronisation der Medien. Multimedia-Präsentationen, bei denen der zeitliche Ablauf vorgegeben ist, nennt man orchestriert.

Damit eine orchestrierte Multimedia-Präsentation nicht wie ein Film abläuft, integrieren Multimedia-Dokumente Interaktionsmöglichkeiten, die es dem Benutzer erlauben, den Ablauf der Präsentationen zu beeinflussen. Beispiele für Benutzerinteraktion sind das Traversieren eines Verweises zu einem anderen Multimedia-Dokument oder die Veränderung der Präsentationsgeschwindigkeit eines Video-Clips.

Die Spezifikation von Hypermedia-Dokumenten erfolgt in einer Dokumentensprache, die Konstrukte, auch Dokumentelemente genannt, zur Beschreibung des Inhalts und der Struktur des Dokuments anbietet. Dokumentelemente besitzen Attribute, die ihre Eigenschaften beschreiben. Jede Hypermedia-Dokumentensprache basiert auf einem Dokumentenmodell, das je nach Einsatzbereich unterschiedlich komplexe und anwendungsspezifische Konstrukte für Medien und Strukturbeschreibung integrieren kann.

In dieser Abhandlung spielt die Vernetzung von Dokumenten eine untergeordnete Rolle, im Mittelpunkt stehen orchestrierte Multimedia-Dokumente. Hierbei werden allgemeine Dokumentenmodelle betrachtet. Das sind Dokumentenmodelle, die für ein großes Spektrum von Anwendungsbereichen, wie Produktpräsentationen, Unterhaltung und Lehr-/Lernsysteme, einsetzbar sind.

In einer verteilten Umgebung werden Multimedia-Dokumente häufig in digitalen Bibliotheken gespeichert. Hierbei kann die strukturelle Beschreibung eines Multimedia-Dokuments getrennt von den eigentlichen Mediendaten gespeichert sein. Zusätzlich kann auch die strukturelle Beschreibung eines logischen Dokuments verteilt gespeichert sein, was jedoch in der Praxis eher selten vorkommt.

Der Nutzer kann nun mittels eines Präsentationsterminals über ein Netzwerk auf Dokumente einer solchen Bibliothek zugreifen und sie mit Hilfe eines geeigneten Präsentationssystems darstellen. Abbildung 1-2 zeigt ein derartiges Szenario. Das größte und sehr populäre verteilte Hypermedia-Dokumentensystem ist das World-Wide-Web (WWW).

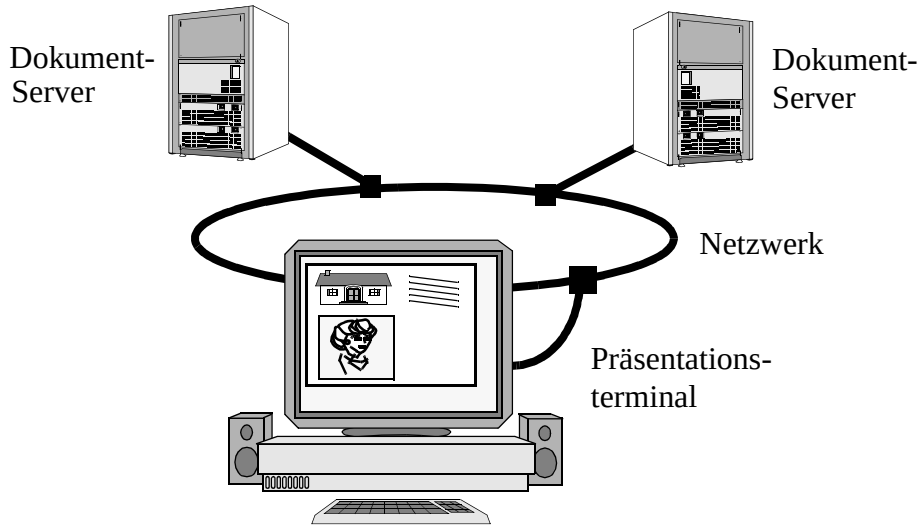


Abb. 1-2 : Verteilte Multimedia-Präsentation

Zur Präsentation eines Multimedia-Dokuments müssen Ressourcen in ausreichender Menge verfügbar sein. Es ist eine entsprechende Bandbreite zwischen den Dokument-Servern und dem Präsentationsterminal erforderlich, um das Dokument zur Präsentation auf das Terminal zu transferieren. Auf dem Präsentationsterminal wird Puffer zur Speicherung des Dokuments benötigt sowie CPU-Rechenzeit zur Interpretation des Dokuments und Generierung der entsprechenden Präsentation. Je nach Art der im Dokument enthaltenen Medien können spezielle Hardware-Komponenten des Präsentationsterminals, wie beispielsweise eine Audio-Karte oder ein Video-Dekoder, benötigt werden. Durch die in Multimedia-Dokumenten integrierten kontinuierlichen Medien kann der Ressourcenbedarf von Multimedia-Präsentationen sehr hoch in Bezug auf Bandbreite, Puffer und CPU-Rechenzeit sein. Sind nicht genügend Ressourcen verfügbar, ist es unmöglich die Präsentation, wie durch das Multimedia-Dokument definiert, zu generieren, was zu Qualitätsverlusten bis zum Abbruch der Präsentation führen kann.

Multimedia-Dokumente, die auf CD-ROM verkauft werden, beseitigen das potentielle Problem der Ressourcenknappheit indem Mindestvoraussetzungen bezüglich der Leistungsfähigkeit des Computers, auf denen sie präsentiert werden können, definiert werden. Für online verfügbare Multimedia-Dokumente ist dieser Ansatz nicht sinnvoll, da er ihre Nutzungsmöglichkeiten zu sehr einschränken würde. Beispielsweise sollte es möglich sein, unabhängig von der Art des verwendeten Präsentationsterminals und des Netzwerks, Dokumente aus Digitalen Bibliotheken zu verwenden. D.h. ein Nutzer sollte mittels eines leistungsschwachen Persönlichen Digi-

talen Assistenten (PDA) über ein schmalbandiges Funknetz genauso wie mit einem leistungsstarken Personal Computer (PC) über ein breitbandiges Netzwerk, wie beispielsweise ATM [LDK94], auf sie zugreifen können.

Dementsprechend sind Ressourcenknappheiten beim Konsumieren von online Multimedia-Dokumenten eher die Regel und nicht die Ausnahme [Bult93]. Dies kann jeder leicht validieren, indem er versucht einen Video-Clip aus dem WWW anzuschauen, der als Datenstrom in Echtzeit übertragen wird. Auch in der Zukunft wird es höchstwahrscheinlich keinen Ressourcenüberfluß bei Multimedia-Präsentationen geben, da mit dem Ausbau von Netzwerken und der Leistungssteigerung von Terminals auch die Nutzeranforderungen steigen, was wiederum zu einem höheren Datenvolumen und damit Ressourcenbedarf führt.

Das Problem der Ressourcenknappheit wird auch durch die Reservierung von Ressourcen nicht grundsätzlich gelöst. Da einerseits Transportprotokolle, wie RSVP [ZDE+93] und ST-II [Topo90], oder Netzwerke, wie ATM, die Garantien für Ressourcen geben können, nicht flächendeckend verfügbar sind. Andererseits selbst wenn eine Ressourcenreservierung möglich ist, können in Abhängigkeit des verwendeten Netzwerks und Terminals sowie des Präsentationszeitpunkts zu wenig freie Ressourcen verfügbar sein, die man für die Dokument-Präsentation reservieren kann. Falls jedoch genügend freie Ressourcen da sind, kann man durch Ressourcenreservierung verhindern, daß durch unterschiedliche Systemauslastungen während einer Präsentation Ressourcenknappheiten auftreten. In der Regel variiert der Ressourcenverbrauch während der Präsentation eines Multimedia-Dokuments in Abhängigkeit der momentan präsentierten Medien. Will man eine effiziente Nutzung von Ressourcen erreichen, sollte man Überreservierungen vermeiden. Ressourcen sollten daher je nach Präsentationsverlauf inkrementell allokieren und freigegeben werden und nicht am Anfang der Präsentation der maximale Bedarf allokiert werden. Eine inkrementelle Allokierung von Ressourcen kann jedoch aufgrund mangelnder Verfügbarkeit fehlschlagen, was wiederum zu einer Ressourcenknappheit führt.

Wenn ein Ressourcenengpaß auftritt, bestehen aus Sicht des Präsentationssystems im wesentlichen drei Reaktionsmöglichkeiten. Die rigorose Möglichkeit ist, die Präsentation zu beenden, sobald festgestellt wird, daß die Präsentation mit den vorhandenen Ressourcen nicht fortgeführt werden kann. Zweitens, man kann den Engpaß einfach ignorieren und akzeptieren, daß die Qualität der Präsentation unkontrolliert sinkt. Dies kann sich beispielsweise dadurch bemerkbar

machen, daß es zu unangenehmen Unterbrechungen beim Ausspielen von Video-Clips und Audio-Sequenzen kommt. Falls entsprechende Effekte massiv über längere Zeit auftreten, kann es auch hier unmöglich sein, die Präsentation fortzuführen, da das Präsentationssystem Verzögerungen und Aussetzer nicht geeignet verarbeiten kann. Zudem wird der Nutzer je nach Leidensfähigkeit die Präsentation bei wiederholten massiven Qualitätseinbrüchen früher oder später selbst beenden. Im Vergleich zum ersten Ansatz werden jedoch Präsentationen mit sporadisch auftretenden Ressourcenengpässen nicht sofort beendet. Die dritte Möglichkeit besteht darin, die Präsentation in einer kontrollierten Weise an die gegebene Ressourcensituation anzupassen, beispielsweise indem zur Senkung des Ressourcenverbrauchs die Präsentation einzelner Medien beendet wird, wodurch die Qualität der übrigen Präsentation dann weiterhin akzeptabel ist. Aus Sicht des Nutzers ist die letzte Möglichkeit die beste, da hier insgesamt die geringsten Qualitätseinbußen zu erwarten sind [Gesc97].

Eine kontrollierte Anpassung des Ressourcenbedarfs einer Präsentation kann nicht beliebig durch den Nutzer oder das Präsentationssystem vorgenommen werden, da ohne Berücksichtigung von Inhalt und Zweck eines Multimedia-Dokuments der inhaltliche Zusammenhang verloren gehen kann, wodurch die Präsentation sinnlos wird. Adaptionen ohne wesentlichen Qualitätsverlust sind nur möglich, wenn die Semantik des Dokuments und der darin vorkommenden Medien beachtet wird. Da die Semantik im allgemeinen nur der Autor eines Dokuments kennt, muß er die Adaptionsmöglichkeiten definieren.

Um Adaptionsmöglichkeiten zu schaffen, könnte der Autor für jede potentielle Ressourcensituation eine entsprechende Version desselben logischen Dokuments erstellen. Dieser Ansatz bringt jedoch, außer dem erheblichen Spezifikationsaufwand, noch weitere Probleme mit sich. Die Vorhersage der potentiellen Ressourcensituationen ist nahezu unmöglich. Selbst wenn die Vorhersage gelingt, entstehen viele Versionen desselben logischen Dokuments. Denkt man einmal an die Verwaltung von Multimedia-Dokumenten in Digitalen Bibliotheken, wird das Dokumenten-Management erheblich aufwendiger.

Ein wesentlich besserer Ansatz ist, nur ein Multimedia-Dokument zu haben und in diesem Dokument alternative Präsentationen zu spezifizieren, so daß während der Präsentation je nach Ressourcensituation geeignete Präsentationsalternativen ausgewählt werden können. Ein Mul-

timedia-Dokument, das diese Eigenschaft besitzt, nennt man adaptiv. Diese Arbeit konzentriert sich auf zeitlich adaptive Multimedia-Dokumente.

Leider ist der Begriff adaptiv oder adaptierbar im Zusammenhang mit Multimedia-Dokumenten mehrfach belegt. Beispielsweise ist in [ROH+97] mit adaptierbar gemeint, daß Dokumente in einer Plattform-unabhängigen Dokumentensprache spezifiziert sind.

1.2 Adaptive Multimedia-Dokumente

Zur Realisierung adaptiver Multimedia-Dokumente werden Dokumentenmodelle benötigt, die es erlauben Präsentationsalternativen zu modellieren. Adaptivität in Multimedia-Dokumenten kann auf zwei Ebenen betrachtet werden, auf Dokumentelementebene und Attributebene. Es gibt bereits Dokumentenmodelle, mit denen man einfache adaptive Dokumente erstellen kann. Die Modelle Firefly [BuZe92, BuZe93] und Mbuild [HSR92, HSR94] bieten Möglichkeiten Adaptivität auf Attributebene zu spezifizieren. Die Modelle CHIMP [CPS96] und MODE [BHL91] bieten zusätzlich noch rudimentäre Möglichkeiten um Adaptivität auf Dokumentelementebene zu spezifizieren. Problematisch bei diesen Ansätzen ist, daß nicht alle möglichen Varianten von Präsentationsalternativen definiert werden können, die Auswahl von Präsentationsalternativen nur unzureichend gesteuert werden kann und keine variablen Abstraktionen angeboten werden, um zu beschreiben, wie die an der Präsentation beteiligten Medien bei Adaptionen manipuliert werden sollen.

Lösungsansätze für diese Problembereiche liefert das in dieser Arbeit präsentierte Dokumentenmodell Tiempo (Temporal Integrated Model to Present Multimedia Objects), das am Institut für Parallele und Verteilte Systeme der Universität Stuttgart entwickelt wurde. Tiempo integriert eine vollständige Menge von Abstraktionen zur Spezifikation von Präsentationsalternativen. Es erlaubt die Spezifikation von Adaptivität auf Dokumentelementebene durch Auswahlgruppen, mit Hilfe derer alternative Dokumentelemente spezifizierbar sind, und auf Attributebene durch Dienstgütebereiche, welche die Spezifikation alternativer Attributwerte ermöglichen. Das Modell erlaubt es, durch Qualitätsmaße und Auswahlpolitiken die Selektion von Präsentationsalternativen detailliert zu steuern. Durch sie kann festgelegt werden, welche Präsentation in einer gewissen Ressourcensituation gewählt werden soll. Um die durch Adaptionen erforderlichen Manipulationen von in einer Präsentation enthaltenen Medien zu model-

lieren, dienen ausdruckstarke Projektionen. Das Tiempo-Modell berücksichtigt somit vielfältige Aspekte von Adaptivität in Multimedia-Dokumenten.

Im Kontext adaptiver Multimedia-Dokumente entstehen neue Problembereiche: Mehrere alternative Präsentationen in einem Dokument führen zu einer höhere Komplexität der Dokumentspezifikation. Es sind daher geeignete Unterstützungstechniken für den Erstellungsprozeß und die Konsistenzsicherung adaptiver Multimedia-Dokumente erforderlich. Die Möglichkeit alternative Präsentationen zu erzeugen, erfordert geeignete Ablaufplanungsverfahren, die bei sich ändernden Ressourcensituationen geeignete Anpassungen durchführen. Diese Abhandlung analysiert diese Problembereiche und präsentiert Vorschläge zu deren Lösung.

1.2.1 Spezifikation adaptiver Multimedia-Dokumente

Multimedia-Dokumente werden häufig durch Werbefachleute oder Pädagogen ohne Programmierkenntnisse erstellt. Es werden daher Autorensysteme benötigt, die eine graphisch interaktive Erstellung von Multimedia-Dokumenten erlauben. Mehrere alternative Präsentationen in einem Dokument führen zu einer höheren Komplexität der Dokumentspezifikation, die in herkömmlichen Dokumentenmodellen nicht vorhanden ist. Daher sind für Autorensysteme von adaptiven Dokumenten neue Konzepte erforderlich, die eine übersichtliche Spezifikation von Präsentationsalternativen ermöglichen. Das in dieser Arbeit vorgestellte Visualisierungs- und Spezifikationskonzept für das Tiempo-Modell kombiniert die Vorzüge der Bildschirm-, Zeitachsen- und Flowchart-basierten Visualisierungsansätze mit objektorientierten Interaktionsparadigmen, um diesem Anspruch zu genügen.

Bei der Erstellung einer Multimedia-Präsentation mit einem Autorensystem muß sichergestellt sein, daß nur konsistente Dokumente entstehen, die keine Fehler und Widersprüche enthalten. Dies gilt insbesondere im Bezug auf den spezifizierten zeitlichen Ablauf der Präsentation. Bei herkömmlichen Autorensystemen wird die Korrektheit der zeitlichen Spezifikation meist dadurch überprüft, daß versucht wird einen Ablaufplan für die Präsentation zu erstellen. Gelingt dies, war die Spezifikation korrekt, ansonsten nicht. Da adaptive Multimedia-Dokumente durch die Präsentationsalternativen jedoch nicht einen, sondern sehr viele unterschiedliche Ablaufpläne besitzen, ist es nicht effizient und teilweise auch nicht möglich zur Konsistenzprüfung Ablaufpläne zu erzeugen.

Das für das Tiempo-Modell entwickelte Konsistenzprüfungsverfahren evaluiert die Konsistenz von Dokumentspezifikationen durch den Einsatz graphenbasierter Algorithmen, was wesentlich effizienter ist. Als Nebeneffekt liefert das Verfahren die vollständig konsistenten Werteintervalle aller zeitlichen Attribute eines adaptiven Dokuments. Das sind die Werteintervalle, deren Werte zu keiner Inkonsistenz führen. Diese Werteintervalle kann man in Autorensystemen anzeigen, wodurch die Spezifikation von Präsentationsalternativen auf Attributebene erheblich vereinfacht wird und Spezifikationsfehler verringert werden.

1.2.2 Adaptive Ablaufplanung

Um eine orchestrierte Präsentation entsprechend einem Multimedia-Dokument durchführen zu können, muß ein Ablaufplan erstellt werden. Ein solcher Ablaufplan beschreibt, wann Medien mit welcher Charakteristik präsentiert werden. Bei der adaptiven Ablaufplanung wird der Ablaufplan unter Berücksichtigung des Ressourcenbedarfs der Präsentation und der verfügbaren Ressourcen erstellt. Dabei werden geeignete Präsentationsalternativen so ausgewählt, daß keine Ressourcenengpässe entstehen. Das adaptive Tiempo-Ablaufplanungsverfahren berücksichtigt durch Beobachtung ermittelte Informationen über Ressourcenanforderungen und verfügbare Ressourcen. Es kann in Systemumgebungen, in denen eine Ressourcenzuteilung nach dem Best-Effort Prinzip oder durch Ressourcenreservierung erfolgt, eingesetzt werden. In dieser Arbeit wird die Best-Effort Zuteilung einer genaueren Betrachtung unterzogen.

Im Vergleich mit existierenden adaptiven Ablaufplanungsverfahren, wie für CHIMP und MODE, ermöglicht das Tiempo-Verfahren eine detailliertere Beschreibung von Ressourcenanforderungen und die Berücksichtigung aller relevanten Ressourcenklassen.

1.2.3 Multimedia-Dokumentensysteme

Durch die stetig wachsende Leistungsfähigkeit von Computern und Netzwerken eröffnen sich ständig neue Möglichkeiten bei der Gestaltung von Multimedia-Präsentationen, zudem stellen Nutzer laufend höhere Anforderungen an das Design und die Gestaltung von Multimedia-Präsentationen. Im Bezug auf Multimedia-Dokumentensysteme ergibt sich damit die Anforderung, daß sie einfach um neue Medientypen und Darstellungsmöglichkeiten erweitert werden können müssen.

Basierend auf den Anforderungen des Tiempo-Modells wurde eine objektorientierte Dokumentensystem-Architektur entwickelt, welche diesen Erweiterungsaspekt berücksichtigt. Entsprechend dieser Architektur werden Spezifikations-, Konsistenzprüfungs- und Ablaufplanungsfunktionalitäten in generischen Präsentationskontrollobjektklassen gekapselt, von denen beliebige medienspezifische Präsentationskontrollobjektklassen abgeleitet werden können, welche Dekodierungs- oder Darstellungsfunktionalitäten beinhalten. Dadurch muß man bei der Erweiterung des Dokumentensystems nur neue Dekodierungs- oder Darstellungsaspekte realisieren, aber keine Erweiterungen am Tiempo-Modell und dessen Algorithmen vornehmen.

Anhand dieser Architektur wurde das Tiempo-Dokumentensystem prototypisch implementiert, das aus einem Dokumenteditor zur graphisch-interaktiven Komposition adaptiver Multimedia-Dokumente entsprechend dem Tiempo-Modell und einem Präsentationssystem zur Erzeugung adaptiver Präsentationen besteht. Mit Hilfe dieses Dokumentensystems wurden die in dieser Arbeit vorgestellten Konzepte und Algorithmen verifiziert.

Das Tiempo-Dokumentensystem integriert keine Funktionalität zur Speicherung oder Verwaltung von Multimedia-Dokumenten. Es ist darauf ausgerichtet mit existierenden Digitalen Bibliotheken und Dokument-Servern zusammenzuarbeiten, welche die standardisierten Kommunikationsprotokolle HTTP [FGM+97], RTSP [SRL97] und RTP [SCF+96] unterstützen.

1.3 Überblick über die Arbeit

In Kapitel 2 wird der Begriff Multimedia-Dokument näher definiert und die Architektur von Multimedia-Dokumenten untersucht. Anhand dieser Analyse werden Möglichkeiten zur Flexibilisierung von Multimedia-Dokumenten abgeleitet und untersucht, inwieweit bereits entsprechende Konzepte existieren und in welchen Bereichen neue Ansätze zu entwickeln sind. In Kapitel 3 wird das adaptive Dokumentenmodell Tiempo vorgestellt, auf das sich die anderen in dieser Arbeit vorgestellten Konzepte beziehen.

Die Konzeption eines Autorensystems, das den Spezifikationsprozeß für adaptive Multimedia-Dokumente geeignet unterstützt, wird in Kapitel 4 vorgestellt. In Kapitel 5 wird ein Konsistenzprüfungsverfahren beschrieben, das in Autorensystemen eingesetzt werden kann, um die fehlerfreie Komposition von Dokumenten zu garantieren.

Mögliche Ansätze zur Ablaufplanung adaptiver Multimedia-Dokumente werden im Kapitel 6 untersucht. Kapitel 7 und 8 stellen den adaptiven Tiempo-Ablaufplanungsalgorithmus vor. In Kapitel 8 werden Meßergebnisse präsentiert, welche die Leistungsfähigkeit der vorgestellten Konzepte zeigen.

Die erweiterbare Architektur des Tiempo-Dokumentensystems wird in Kapitel 9 vorgestellt. Die Abhandlung schließt mit einer Zusammenfassung und einem Ausblick.

2 Adaptivität multimedialer Dokumente

In diesem Kapitel wird die Adaptivität von Multimedia-Dokumenten und die Modellierung von Präsentationsalternativen in Dokumentenmodellen näher betrachtet. Dazu wird zunächst die grundlegende Architektur von Multimedia-Dokumenten analysiert. In Abschnitt 2.2 wird untersucht in welcher Weise Multimedia-Dokumente flexibel sein können, so daß Adaptionen möglich werden. Danach wird analysiert inwieweit existierende Dokumentenmodelle die Spezifikation adaptiver Multimedia-Dokumente unterstützen und in welchen Bereichen neue Konzepte erforderlich sind.

2.1 Architektur von Multimedia-Dokumenten

Ein Multimedia-Dokument kann wie folgt definiert werden:

Def.: Ein **Multimedia-Dokument** ist eine Menge strukturierter Informationen, die aus unterschiedlichen Medien besteht, zur Aufnahme durch den Menschen bestimmt ist und einer Verarbeitung durch Computer zugänglich ist [Stei93b].

Entsprechend dieser Definition besteht ein Multimedia-Dokument aus Medien und Strukturinformation, welche die Medien in Beziehung setzt. Beispiele für solche Beziehungen sind räumliche, zeitliche oder interaktive Beziehungen.

2.1.1 Medienklassen

Die Medien in Multimedia-Dokumenten können unterschiedlich klassifiziert werden (Abbildung 2-1). Es gibt drei Grundklassen von Medien: diskrete, kontinuierliche und zusammengesetzte Medien.

Diskrete Medien haben keinen inhärenten Zeitbezug. Beispiele für diskrete Medien sind Texte, Bilder, Graphiken. **Kontinuierliche Medien** besitzen einen inhärenten Zeitbezug. Zu dieser Klasse gehören Audios, Videos und Animationen. Kontinuierliche Medien bestehen aus einer vollständig geordneten Menge einzelner Dateneinheiten. Ein Video-Clip besteht beispielsweise aus einer geordneten Menge von Einzelbildern (Frames) und ein Musikstück aus einer geord-

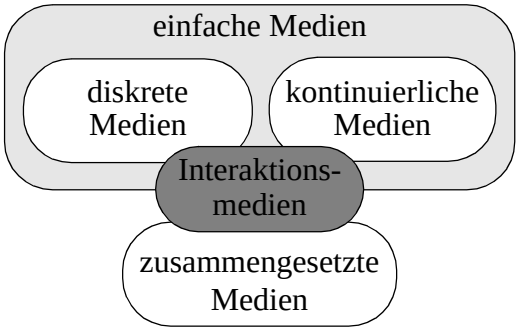


Abb. 2-1 : Medienklassen

neten Menge von Samples. Ein kontinuierliches Medium besitzt eine natürliche Rate r_0, welche bei der Erzeugung des Mediums festgelegt wird, sei es durch Aufnahme mit einer Video-Kamera oder einem digitalen Audio-Rekorder oder durch synthetisches Erzeugen mittels entsprechender Computer-Anwendungen. Die mit einem Medium assoziierte natürliche Periode p_0 ist reziprok zu r_0: $p_0 = 1/r_0$ und beschreibt wie lange jede Dateneinheit des Mediums ausgespielt wird. Aus der Anzahl n der Dateneinheiten eines Mediums und der Rate r_0 läßt sich seine natürliche Präsentationslänge als $l_0 = n/r_0$ berechnen. Diskrete und kontinuierliche Medien werden als **einfache Medien** bezeichnet.

Einfache Dokumentenmodelle unterstützen nur eine Art **zusammengesetzter Medien,** nämlich Dokumente. Große Multimedia-Dokumente können jedoch viele einfache Medien und dementsprechend komplexe Strukturinformationen enthalten. Um die Übersichtlichkeit der Dokumentspezifikation zu verbessern, bieten komplexere Dokumentenmodelle daher weitere Strukturierungs- oder Gruppierungsmöglichkeiten an. Beispielsweise haben Dokumente, insbesondere mit hohen Textanteilen, logisch oft eine hierarchische Struktur. Deshalb können in vielen Modellen Medien hierarchisch geschachtelt werden, das heißt, es können zusammengesetzte Medien gebildet werden, die Dokumentteile repräsentieren und aus einfachen und anderen zusammengesetzten Medien bestehen.

Jede der vorgestellten Medienklassen kann sensitiv für eine Benutzerinteraktion sein. Primär interagiert der Benutzer mit diskreten Medien wie z.B. Knöpfen, Schiebereglern oder Textteilen eines Hypertexts. Aber auch zusammengesetzte Medien können sensitiv für Benutzerinteraktionen sein, beispielsweise wenn sie mit manipulierbaren Fenstern auf dem Bildschirm assoziiert werden. Gleiches gilt für kontinuierliche Medien wie Videos oder Animationen, beispielsweise

indem Objekte in einem Film sensitiv für Mausklicks sind und es so ermöglicht wird, Zusatzinformationen abzurufen. Falls ein Medium Benutzerinteraktionen akzeptieren kann, wird es als **Interaktionsmedium** bezeichnet.

Medien werden in Dokumentensprachen durch Konstrukte repräsentiert, die man Medienelemente nennt. Den Medienelementen zugeordnete Attribute beschreiben die Eigenschaften der Medien. Hierbei kann man zwischen temporalen Attributen, welche die zeitlichen Eigenschaften beschreiben, räumlichen Attributen, welche die räumlichen Aspekte des Mediums beschreiben, gestalterischen Attributen, welche die Darstellung des Mediums definieren, und Inhaltsattributen, welche die Mediendaten beinhalten oder referenzieren, unterscheiden.

2.1.2 Räumliche Layout-Definition

Unter der räumlichen Anordnung von Medien versteht man, diese in einem Koordinatensystem zu plazieren, das mit dem Terminal-Bildschirm assoziiert wird. Ein solches Koordinatensystem wird in der Regel dreidimensional sein. Es besteht aus einer horizontalen und vertikalen Achse, welche mit dem zweidimensionalen Bildschirm assoziiert werden, und einer Achse, die zur Spezifizierung der Überlappungen von Medien auf dem Bildschirm dient. Entsprechend der Positionierung im Koordinatensystem erscheinen die Medien bei der Präsentation des Dokuments auf dem Bildschirm. Die Anordnung der Medien im Koordinatensystem bezeichnet man als **räumliches Layout**.

In den meisten existierenden Dokumentenmodellen wird die räumliche Ausdehnung eines Mediums in räumlichen Attributen durch seine Breite und Höhe beschrieben. Um die Positionierung von Medien im räumlichen Koordinatensystem zu definieren, gibt es im Prinzip zwei Möglichkeiten. Medien können absolut zum Ursprung des Koordinatensystems oder relativ zueinander positioniert werden [PAP95]. Bei einer absoluten Positionierung besitzt das Medienelement räumliche Attribute, die den horizontalen und vertikalen Abstand zum Ursprung festlegen, sowie ein Attribut, welches festlegt, wie weit das Medium im Hintergrund liegt. Bei einer relativen Positionierung sind Operatoren erforderlich, mit denen Medienelemente geeignet miteinander verknüpft werden können, um Abstände zwischen Medien zu spezifizieren.

2.1.3 Zeitliche Layout-Definition

Die zeitliche Anordnung bzw. das **zeitliche Layout** definiert, wie Medien während der Präsentation synchronisiert werden. Synchronisation kann auf zwei Ebenen betrachtet werden. Die **Synchronisation auf Medienebene** beschreibt, wann die Präsentation von Medien beginnt und endet. Die **Synchronisation auf Stromebene** definiert, welche Zeitabstände zwischen den Dateneinheiten kontinuierlicher Medien einzuhalten sind, wenn sie entsprechend der Synchronisation auf Medienebene ausgespielt werden [Stei93b, Stei94].

2.1.3.1 Stromsynchronisation

Wird ein kontinuierliches Medium präsentiert, werden seine Dateneinheiten nacheinander in Form eines Datenstroms ausgespielt. Hierbei beschreiben zeitliche Attribute die Präsentationsrate, d.h. mit welcher Periode Dateneinheiten ausgespielt werden, und die Präsentationsgeschwindigkeit, d.h. welche Dateneinheiten ausgespielt werden. Die Präsentationsrate muß hierbei nicht gleich der natürlichen Rate des Mediums sein.

Um sicherzustellen, daß das Ausspielen von Datenströmen kontinuierlich durchgeführt wird, sind Stromsynchronisationsverfahren erforderlich. Hierbei kann man zwischen Intra- und Interstromsynchronisation unterscheiden:

- Die **Intrastromsynchronisation** stellt sicher, daß die Dateneinheiten eines Mediums periodisch ausgespielt werden. Aufgrund von Systemeinflüssen, wie Prozeßwechsel und kurzfristigen Wartezeiten beim Zugriff auf Systemkomponenten, ist das absolut periodische Ausspielen eines Datenstroms in der Praxis nicht möglich. Es tritt ein sogenannter **Präsentations-Jitter** auf, d.h. Dateneinheiten werden etwas früher oder später ausgespielt. Aufgabe der Intrastromsynchronisation ist es, den Jitter unter einem vorgegebenen Wert zu halten [Stei90, Stei92].

- Die **Interstromsynchronisation** stellt sicher, daß beim gleichzeitigen Ausspielen mehrerer Datenströme deren Präsentation kontinuierlich fortschreitet, so daß kein Medium schneller oder langsamer ausgespielt wird als das andere. Da ein absolut gleichförmiges Ausspielen aus den oben genannten Gründen auch hier so gut wie unmöglich ist, tritt ein sogenannter **Skew** auf, der beschreibt wie weit die Präsentation eines Datenstroms gegen-

über den anderen Datenströmen voraus oder hinterher ist. Aufgabe der Interstromsynchronisation ist es, den Skew unter einem vorgegebenen Wert zu halten [LiKo92].

Dokumentenmodelle können es erlauben, daß Jitter und Skew durch den Autor spezifiziert werden oder implizit sinnvolle Jitter- und Skew-Werte verwenden, beispielsweise entsprechend der in [StEn93] beschriebenen Untersuchung von akzeptablen Jitter- und Skew-Werten.

2.1.3.2 Mediensynchronisation

Bei der Definition der Synchronisation auf Medienebene gibt es zwei grundlegende Ansätze:

- Beim **Ereignis-basierten Ansatz** werden Medien zeitlich durch Ereignisse beschrieben. Ein Startereignis definiert, wann die Präsentation des Mediums beginnt, und ein Endereignis definiert, wann die Präsentation des Mediums endet. Zur Beschreibung des zeitlichen Layouts werden diese Ereignisse auf einer Zeitachse, die den Ablauf der Dokumentpräsentation beschreibt, absolut oder relativ zueinander angeordnet.

- Beim sogenannten **Intervall-basierten Ansatz** werden Medien durch ihre Präsentationsdauer in Form eines Intervalls beschrieben. Zur Spezifikation des temporalen Layouts werden diese Intervalle absolut oder relativ zueinander positioniert. Bei absoluter Positionierung besitzen die Medienelemente ein Attribut, das den Abstand zum Nullpunkt der Zeitachse angibt. Bei relativer Positionierung sind Operatoren erforderlich, um die Beziehungen zwischen Ereignissen oder Intervallen zu definieren.

Basierend auf diesen beiden grundlegenden Ansätzen wurden vielfältige Zeitmodelle und Synchronisationskonzepte für Multimedia-Dokumente entwickelt, die sich grob in drei Klassen einteilen lassen. Es gibt unterschiedliche Variationen von Zeitachsen-basierten Verfahren [Bruc72, Appl91, Drap93, Hosc98, HSR94, HyTi92, KrCa92, MHEG94, RJM+93, ViKa86]. In diesen Modellen werden Medien explizit auf Zeitachsen angeordnet. Ein weiterer beliebter Ansatz ist es, Synchronisationsaspekte mit Hilfe von Petri-Netzen zu beschreiben [CoRo83, LiGh90, QWG93, PrRa93, SDL+96]. Petri-Netz Ansätze erlauben es, Synchronisationsaspekte auf Strom- und Medienebene zu modellieren. Des weiteren gibt es Vorschläge, die zeitlichen Beziehungen zwischen Medien durch spezielle Objekte, Abstraktionen oder Operatoren zu modellieren [BHL91, BuZe93, Bole95, CaHa74, CPS96, DMP89, Gibb91, Hoar85, Hoep91, KeDu96,

SKD96, SLF+96, WaRo94]. Ein detaillierter Überblick über Medien-Synchronisationskonzepte in Multimedia-Dokumenten und ihre Eigenschaften ist in [PeLi96] zu finden.

2.1.4 Interaktionen

Interaktionsmöglichkeiten erlauben es dem Benutzer auf die Präsentation eines Multimedia-Dokuments Einfluß zu nehmen. Benutzerinteraktionen in Multimedia-Dokumenten kann man wie folgt klassifizieren:

- **Räumliche Interaktionen** beeinflussen die räumliche Darstellung von Medien. Hierzu zählt die Veränderung der Mediengröße, das Verschieben von Medien oder das Verändern der Überlappung von Medien.

- **Gestalterische Interaktionen** beeinflussen die Gestaltung von Medien. Hierzu gehört beispielsweise die Veränderung der Farbe eines dargestellten Mediums.

- **Zeitliche Interaktionen** beeinflussen den Ablauf der Dokumentpräsentation, wie beispielsweise durch das Traversieren eines Hyperlinks oder die Veränderung der Präsentationsgeschwindigkeit eines Video-Clips.

- **Stukturverändernde Interaktionen** erlauben es, das Dokument selbst zu verändern, beispielsweise durch Hinzufügen oder Löschen von Medienelementen.

Interaktionen unterschiedlicher Klassen können kombiniert auftreten. Insbesondere räumliche und gestalterische Interaktionen treten oft gemeinsam auf, beispielsweise wenn sich beim in den Vordergrund holen eines Fensters dessen Rahmenfarbe ändert.

Eine Interaktion ist durch ein Interaktionsmedium, welches die zur Auslösung erforderliche Benutzeraktion registriert, die beeinflußten Dokumentelemente und die Beziehung, welche definiert, was beim Auslösen der Interaktion mit den beeinflußten Dokumentelementen passieren soll, charakterisiert. Bei manchen Interaktionen sind Interaktionsmedium und beeinflußtes Dokumentelement identisch. Dokumentenmodelle bieten spezielle Elemente oder Operatoren an, um die erforderlichen Beziehungen zu beschreiben, wenn Interaktionsmedien und beeinflußte Dokumentelemente nicht identisch sind.

2.2 Adaptivität von Multimedia-Dokumenten

Damit eine Multimedia-Präsentation an unterschiedliche Ressourcensituationen angepaßt werden kann, sind flexible Dokumentspezifikationen erforderlich. Flexible Dokumentspezifikationen beinhalten Freiheitsgrade, so daß je nach Ressourcensituation eine geeignete Präsentation zusammengestellt werden kann. Mit anderen Worten, eine flexible Dokumentspezifikation definiert nicht eine Präsentation, sondern eine Menge alternativer Präsentationen. Damit flexible Dokumentspezifikationen erstellt werden können, muß das zugrundeliegende Dokumentenmodell entsprechende Konzepte zur Beschreibung von Flexibilität integrieren. Auf der Basis der in Abschnitt 2.1 beschriebenen Dokumentenarchitektur ist Flexibilität auf Attributebene und Dokumentelementebene möglich.

2.2.1 Flexibilität auf Dokumentelementebene

Dokumente, die Flexibilität auf Dokumentelementebene integrieren, beinhalten alternative Medienelemente, welche dieselben oder ähnliche Informationen in unterschiedlicher Form repräsentieren. In Abhängigkeit der Ressourcensituation kann während der Präsentation eine geeignete Alternative ausgewählt und präsentiert werden. Beispielsweise kann die Information über ein technisches Thema mehrere alternative Darstellungen haben, wie eine Erklärung in Form einer digitalen Sprachsequenz, einer textuellen Beschreibung oder einer Animation. Da verschiedene Informationsformen im allgemeinen unterschiedliche Ressourcenanforderungen haben, können somit unterschiedliche Ressourcensituationen während der Präsentation gehandhabt werden.

Flexible Dokumentenmodelle müssen nicht nur einfache Präsentationsalternativen beinhalten können. Falls die Definition von Präsentationsalternativen erfordert, daß mehrere Medien in unterschiedlicher Art in Beziehung gesetzt werden, sind zusammengesetzte Medienelemente zur Spezifikation alternativer Präsentationen erforderlich.

Flexibilität auf Dokumentelementebene ist nicht auf Medienelemente beschränkt. Dokumente können ebenso vollkommen unterschiedliche zeitliche Anordnungen zwischen denselben Medien enthalten [KeLo91, vBee92]. Beispielsweise macht es oft keinen Unterschied, ob eine Werbeanimation vor oder nach einem spezifischen Medium präsentiert wird, solange sie nur

irgendwann in der gesamten Präsentation vorkommt. Durch alternative Anordnungen entstehen unterschiedliche Überlappungen von Medien in der Präsentation, wodurch auch unterschiedliche Ressourcenanforderungen gegeben sind. Werden Medien in Dokumentenmodellen zeitlich absolut positioniert, kann die Spezifikation alternativer Anordnungen nur mittels alternativer Medienelemente erfolgen. Unterstützt ein Dokumentenmodell Operatoren zur relativen zeitlichen Positionierung, können auch diese Operatoren als Alternativen definiert werden.

Die Definition unterschiedlicher räumlicher Anordnungen von Medien hat keinen Einfluß auf den Ressourcenverbrauch, da unabhängig von der Anordnung immer dieselben Mediendaten mit denselben Charakteristika ausgespielt werden müssen.

Oft enthalten Multimedia-Präsentationen Medien, die wenig zum eigentlichen Informationsgehalt beitragen und lediglich zur Aufheiterung des Nutzers oder zur gestalterischen Verschönerung dienen, wie beispielsweise eine Animation des Firmenlogos im Hintergrund. Das Ausspielen solcher Information könnte beendet oder erst gar nicht begonnen werden, wenn nicht genügend Ressourcen zur Verfügung stehen.

Die vorgestellten Flexibilisierungsmöglichkeiten auf Dokumentelementebene können nicht automatisch in Präsentationen eingefügt werden, da es vom Inhalt und Zweck eines Multimedia-Dokuments abhängt, ob eine alternative Präsentation möglich ist oder nicht. Da dieses Wissen jedoch nur der Autor eines Multimedia-Dokuments hat, muß dieser auch die Flexibilität explizit spezifizieren. Somit müssen Dokumentenmodell und -sprache geeignete Konzepte und Konstrukte zur Spezifikation von Flexibilität auf Dokumentelementebene anbieten.

2.2.2 Flexibilität auf Attributebene

Dokumente mit Flexibilität auf Attributebene erlauben alternative Attributwerte für Dokumentelemente. Dadurch beschreiben diese Dokumente alternative Präsentationen, auch wenn sie keine Flexibilität auf Dokumentelementebene unterstützen.

Auf den Ressourcenbedarf haben Attribute Einfluß, welche die räumliche Ausdehnung von Medienelementen beschreiben. Wenn beispielsweise räumlich nur ein Ausschnitt eines Videos dargestellt wird oder mit einer geringeren räumlichen Auflösung ausgespielt wird, können

Bandbreite und eventuell auch CPU-Ressourcen zum Dekomprimieren auf dem Präsentationsterminal eingespart werden.

Betrachtet man Komprimierungsstandards, wie beispielsweise Motion-JPEG [BFF+96] oder MPEG [Gall91], kann man Ressourcen, wie Bandbreite und Speicher, einsparen, indem man die Komprimierung erhöht. Medienelementen können Attribute zugeordnet werden, die dies beschreiben.

Attribute, die zeitliche Aspekte von Medienelementen beschreiben, beeinflussen ebenfalls den Ressourcenbedarf der Multimedia-Präsentation. Wird beispielsweise die Präsentationsgeschwindigkeit variabel spezifiziert, kann das Präsentationssystem bei Ressourcenknappheit Dateneinheiten langsamer ausspielen, wodurch pro Zeiteinheit weniger Dateneinheiten benötigt werden und dekomprimiert werden müssen. Die Veränderung der Präsentationsgeschwindigkeit bedingt, daß sich die Präsentationsdauer eines Medienobjekts ändert. Für Informationen wie ein animiertes Firmenlogo im Hintergrund wird dies keine große Rolle spielen. Bei einigen Medienarten ist die Veränderung der Präsentationsgeschwindigkeit jedoch nicht möglich. Viele Audiokarten erlauben beispielsweise nur wenige fixe Ausspielgeschwindigkeiten.

Soll sich die Präsentationsgeschwindigkeit nicht ändern, kann die Datenrate eines Mediums skaliert werden, um den Ressourcenbedarf anzupassen. Da dadurch nach einem regelmäßigen Schema Dateneinheiten übersprungen werden, müssen pro Zeiteinheit ebenfalls weniger Dateneinheiten ausgespielt werden. Inwieweit die Veränderung der Datenrate eines kontinuierlichen Mediums möglich ist, hängt von dessen Komprimierung ab. Ist jede Dateneinheit des Mediums unabhängig von den anderen Dateneinheiten komprimiert, wie bei Motion-JPEG oder WAV ist eine beliebige Skalierung möglich. Ist das Medium aber mit einem Verfahren wie MPEG, H.261 [Liou91] oder MP3 komprimiert, bei dem eine komprimierte Dateneinheit von den davor- oder dahinterliegenden Dateneinheiten abhängen kann, ist dies nicht möglich. Eine Skalierung ist dann nur in diskreten Stufen möglich, da sonst Dateneinheit nicht dekomprimiert werden können.

Durch Flexibilisierung von Skew- und Jitter-Attributen kann der Bedarf an Puffer, der zur Kompensation von Skew und Jitter benötigt wird, gesenkt werden.

Eine weitere Flexibilisierung ist im Hinblick auf die Präsentationsdauer eines Mediums möglich. Dadurch wird es beispielsweise möglich, die Präsentation eines Videos früher zu beenden und den Abspann nicht mehr zu präsentieren, wenn die Ressourcen knapp werden.

Integriert ein Dokumentenmodell Operatoren zur Spezifikation des zeitlichen Layouts, ist Flexibilität auf Attributebene auch für diese Operatoren sinnvoll. Erlauben es die Operatoren nämlich variable Verzögerungen zwischen Medien zu definieren, können kurzzeitige Ressourcenengpässe durch Verschieben von Startzeitpunkten der betroffenen Medien überbrückt werden. Im Vergleich dazu hat Flexibilität auf Attributebene bei Operatoren zur Definition des räumlichen Layouts keinen Einfluß auf den Ressourcenverbrauch.

Im Rest dieser Arbeit wird Flexibilität auf Attributebene nur für Attribute, die zeitliche Aspekte beschreiben, betrachtet.

2.2.3 Adaptionskontrolle

Flexible Multimedia-Dokumente versetzen ein Präsentationssystem in die Lage zwischen Präsentationsalternativen auszuwählen, um so auf variierende Ressourcensituationen kontrolliert reagieren zu können. In einer konkreten Ressourcensituation kann es mehrere Möglichkeiten zur Adaption einer Präsentation geben. Beispielsweise kann es in einem Dokument mit Flexibilität auf Dokumentelementebene möglich sein, entweder Untertitel oder eine Sprachsequenz als Erläuterung zu einer Animation zu präsentieren. In einem Dokument mit Flexibilität auf Attributebene kann es möglich sein, entweder die Ausspielgeschwindigkeit eines Videos oder einer Animation zu reduzieren. Da es nur selten offensichtlich ist, welche Präsentationsalternative mit einer besseren Präsentationsqualität assoziiert wird, ist ein objektives **Qualitätsmaß** erforderlich, mittels dem der Autor spezifizieren kann, welche Präsentationsqualität mit welcher Präsentationsalternative korreliert ist. Ein Beispiel für ein solches Qualitätsmaß sind Prioritäten, die Präsentationsalternativen zugewiesen werden und definieren welche Alternativen bevorzugt werden sollen.

Im Hinblick auf den Ablauf einer Präsentation muß definiert werden, wann und unter welchen Umständen das Präsentationssystem eine bestimmte Präsentation aus der Menge der Alternativen wählen darf. Zum Beispiel kann es je nach Kontext und Inhalt einer Präsentation manchmal

gewünscht sein, daß das Präsentationssystem mit dem Ausspielen von Untertiteln fortfährt, wenn das Ausspielen einer Sprachsequenz aufgrund einer Ressourcenknappheit nicht mehr möglich ist. In einem anderen Multimedia-Dokument kann dies aber gar nicht gewünscht sein.

In interaktiven Dokumenten kann der Benutzer häufig von einem Teil der Präsentation in einen anderen springen. Somit könnte er auch zu einem Teil des Dokuments springen, der bereits einmal präsentiert wurde, wodurch dieselben Dokumentteile bzw. Medien mehr als einmal präsentiert werden. In solchen Szenarien kann es gewünscht sein, daß Dokumentteile mit Präsentationsalternativen bei jeder Präsentation dieselben Medien und zeitlichen Anordnung enthalten, so daß sich deren Präsentation grundsätzlich nicht verändert.

Für die Flexibilität auf Attributebene gibt es vergleichbare Szenarien. Zum Beispiel, falls das kontinuierliche Ausspielen eines Mediums ein Qualitätskriterium ist, weil es wichtig für die Vermittlung des Inhalts ist, sollte seine Präsentationsrate während des Ausspielens nicht geändert werden. Es kann dennoch erlaubt sein, vorab eine bestimmte Präsentationsrate zu wählen, mit der das Medium ausgespielt wird. Wenn Dokumentteile wiederholt präsentiert werden können, kann es manchmal gewünscht sein, daß die Medien bei jedem Durchlauf einige identische Eigenschaften haben, wie beispielsweise immer die gleiche Präsentationsdauer.

Um einen Autor in die Lage zu versetzen, darauf Einfluß zu nehmen, wann das Präsentationssystem die Möglichkeit hat Präsentationsalternativen auszuwählen und zwischen Präsentationsalternativen zu wechseln, können Dokumentenmodelle **Auswahlpolitiken** anbieten.

2.3 Existierende adaptive Dokumentenmodelle

Es wurden zahlreiche Dokumentenmodelle für orchestrierte Multimedia-Dokumente entwikkelt. In diesem Abschnitt werden diejenigen Dokumentenmodelle analysiert, die adaptive Spezifikationen unterstützen.

2.3.1 Mbuild

Das Autorensystem Mbuild [HSR92, HSR94] verwendet einen Zeitachsen-basierten Ansatz. In Mbuild werden Medien in mehreren Spuren (Tracks) auf einer Zeitachse positioniert. Verzöge-

rungen zwischen Medien in einer Spur werden durch sogenannten Temporal Glue modelliert. Temporal Glue verhält sich wie der von T_EX [KnPl84] bekannte räumliche Glue: Er kann sich dehnen oder zusammengestaucht werden. Auf diese Weise repräsentiert Temporal Glue flexible Verzögerungen. Parameter dienen zur Kontrolle der Flexibilität. Im Mbuild-System können auch Medien dehnbar und stauchbar spezifiziert werden. Interaktion auf Medienebene wird durch dieses System nicht unterstützt. Abbildung 2-2 zeigt eine Spezifikation mit Temporal Glue. Das Beispiel enthält zwei Spuren, die eine Spur besteht aus einem Audio und die zweite Spur beginnt mit einem Video. Nach Ende des Videos folgt nach einer variablen Verzögerung, die durch ein Glue-Objekt definiert wird, eine Animation.

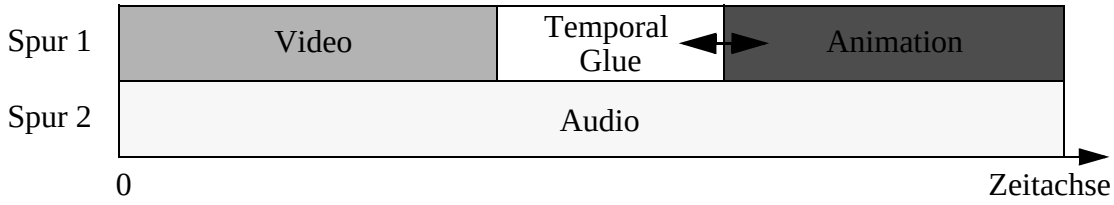


Abb. 2-2 : Spezifikation mit Temporal Glue

Das Präsentationssystem von Mbuild nützt die Flexibilität des Temporal Glue, um das optimale zeitliche Layout einer Präsentation zu bestimmen. Ein zeitliches Layout ist als optimal definiert, wenn alle flexiblen Dauern in einer Spur die gleiche Ausdehnung haben, relativ zur spezifizierten Dehnbarkeit und Stauchbarkeit. Das Präsentationssystem berücksichtigt bei der Bestimmung des optimalen zeitlichen Layouts jedoch nicht die Ressourcensituation. Dementsprechend vereinfacht sich durch die Flexibilität lediglich die Spezifikation von Dokumenten.

2.3.2 Firefly

Im Multimedia-Dokumentensystem Firefly [BuZe92, BuZe93] wird ein Medium durch ein Startereignis und ein Endereignis beschrieben. Ereignisse, die zwischen Start- und Endereignis liegen, werden interne Ereignisse genannt. Interne Ereignisse markieren bestimmte Zeitpunkte, die mit Ereignissen anderer Medien in zeitliche Beziehung gesetzt werden können. Eine weitere Klasse von Ereignissen sind asynchrone Ereignisse, die es erlauben, zeitlich unbestimmtes Verhalten wie Benutzerinteraktionen zu modellieren. Abbildung 2-3 zeigt eine graphische Darstel-

lung dieser Modellierungsmethode, wie sie im Autorensystem des Firefly-Systems zum Einsatz kommt.

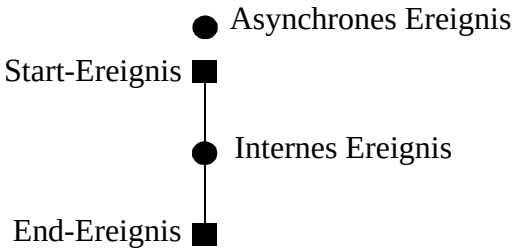


Abb. 2-3 : Modellierung von Medien in Firefly

Verzögerungen bzw. Abstände zwischen Ereignissen in einem Medium werden durch einen minimalen, einen optimalen und einen maximalen Wert beschrieben. Zusätzlich werden Kosten für das Abweichen vom optimalen Wert spezifiziert, und zwar für das Stauchen eines Abstands in Richtung des minimalen Werts und das Dehnen eines Abstands in Richtung des maximalen Werts. Das Firefly-Modell bietet die Möglichkeit, flexible zeitliche Beziehungen zwischen Ereignissen in unterschiedlichen Medien zu definieren. Es kann spezifiziert werden, daß Ereignisse gleichzeitig auftreten müssen oder aber, daß ein flexibler Abstand zwischen ihnen eingehalten werden soll. Es ist jedoch nicht möglich flexiblen Verzögerungen Kosten zuzuweisen. Firefly bietet einfache Knopf-orientierte Nutzerinteraktionen. Es integriert jedoch keine Abstraktionen für zusammengesetzte Medien, außer Dokumente. Aufgrund dieser fehlenden Strukturierungsmöglichkeiten ist die Handhabung größerer Spezifikationen umständlich.

Abbildung 2-4 zeigt eine Dokumentspezifikation entsprechend dem Firefly-Modell. In der Spezifikation stellen der *Term Index* und die *Electricity & Magnetism Lecture* die Hauptpräsentation dar. Die Dauer der *Term Index* Präsentation ist flexibel. Sie kann zwischen 15 und 25 Sekunden liegen, die optimale Dauer ist 20 Sekunden. Die Präsentation der *Electricity & Magnetism Lecture* beginnt 10 Sekunden nach dem *Term Index* und soll 30 Sekunden nach ihm beendet werden. Die Dauer der *Electricity & Magnetism Lecture* ist ebenfalls flexibel spezifiziert. Das asynchrone Ereignis über dem *Term Index* ermöglicht es, eine *Electrolyte Definition* mittels einer Interaktion abzurufen. Diese wird dann für 10 Sekunden angezeigt.

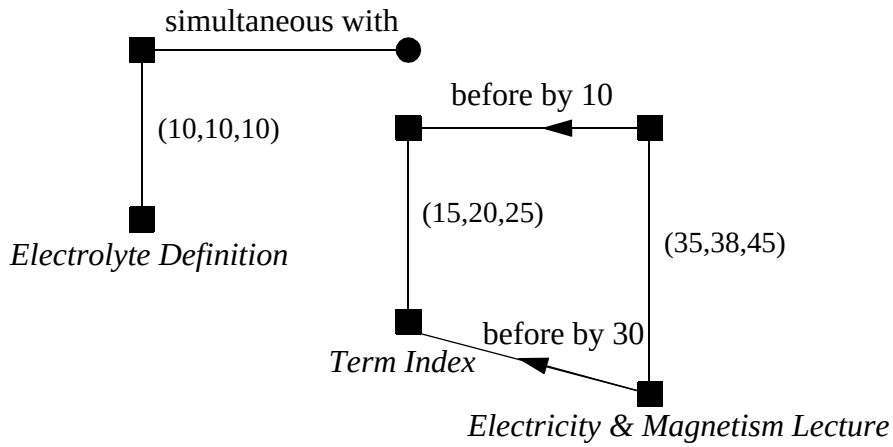


Abb. 2-4 : Firefly-Spezifikationsbeispiel

In Firefly ist das Layout einer Präsentation optimal, wenn die Summe der für das Stauchen oder Dehnen von Medien entstehenden Kosten minimal ist. Wären in Abbildung 2-4 beispielsweise die Kosten für das Stauchen der Dauer des *Term Index* geringer als für das Stauchen der Dauer der *Electicity & Magnetism Lecture*, dann würde Firefly eine Dauer von 38 Sekunden für die *Electricity & Magnetism Lecture* ansetzen und eine Dauer von 18 Sekunden für den *Term Index*. Das Ziel beim Design von Firefly war es, die zeitliche Spezifikation von Multimedia-Dokumenten zu vereinfachen. Deshalb wird im Firefly-System bei der Bestimmung des optimalen zeitlichen Layouts einer Präsentation die Ressourcensituation nicht berücksichtigt.

Ein im Vergleich zu Firefly geringfügig erweiterter Ansatz, der es erlaubt, eine Präsentation einer bestimmten Länge aus einer flexiblen Spezifikation zu generieren, ist in [KiSo95] zu finden. Dieser Ansatz berücksichtigt jedoch, genau wie Firefly, keine Ressourceninformationen.

2.3.3 CHIMP

In CHIMP [CPS96] wird das zeitliche Layout einer Präsentation flexibel mittels sogenannter Difference Constraints modelliert. Difference Constraints sind lineare Bedingungen der Form:

$$x_1 - x_2 \leq b$$

Verwendet man Variablen zur Beschreibung von Präsentationsereignissen (Startzeitpunkt *st* und Endzeitpunkt *et* eines Mediums), dann können Difference Constraints zur Beschreibung flexibler zeitlicher Abstände zwischen diesen Ereignissen verwendet werden. Gibt es beispiels-

weise zwei Medien o_1 und o_2 mit Dauern von 40 und 50 Sekunden und man will, daß o_2 nach o_1 präsentiert wird und die Präsentation von o_2 innerhalb von 10 Sekunden, nachdem o_1 beendet wurde, beginnen soll, kann dies durch folgende Difference Constraints beschrieben werden:

$$st(o_1) - et(o_1) \leq -50 \qquad et(o_1) - st(o_1) \leq 50$$
$$st(o_2) - et(o_2) \leq -40 \qquad et(o_2) - st(o_2) \leq 40$$
$$st(o_2) - et(o_1) \leq 10 \qquad et(o_1) - st(o_2) \leq 0$$

Die Beschreibung des zeitlichen Layouts einer Präsentation durch Difference Constraints ermöglicht es, eine Spezifikation als gerichteten Graph zu repräsentieren. Auf diesen Graph kann ein Kürzester-Pfad Algorithmus angewendet werden, um einen Ablaufplan für das Dokument zu erzeugen. CHIMP unterstützt keine Benutzerinteraktion und keine zusammengesetzten Medien.

CHIMP erlaubt die Spezifikation alternativer Bedingungen und die Zuweisung von Prioritäten zu Bedingungen, wodurch adaptive Spezifikationen ermöglicht werden. Nehmen wir beispielsweise an, es sind zwei Medien o_1 und o_2 mit den Dauern 30 und 20 Sekunden gegeben, deren Präsentation sich nicht überlappen soll. Man wünscht, daß die Verzögerung zwischen o_1 und o_2 nicht größer als 5 Sekunden ist und das o_1 vorzugsweise vor o_2 präsentiert werden soll, es aber auch möglich sein soll o_1 nach o_2 zu präsentieren. Dann kann dies durch die folgenden Bedingungen beschrieben werden:

$st(o_1) - et(o_1) \leq -30$		$et(o_1) - st(o_1) \leq 30$	
$st(o_2) - et(o_2) \leq -20$		$et(o_2) - st(o_2) \leq 20$	
$st(o_2) - et(o_1) \leq 5$	Priorität: 80	$et(o_1) - st(o_2) \leq 0$	Priorität: 80
$st(o_1) - et(o_2) \leq 5$	Priorität: 40	$et(o_2) - st(o_1) \leq 0$	Priorität: 40

Um zu spezifizieren, daß o_1 vor oder nach o_2 präsentiert werden kann, führt man Bedingungen ein, die beide Anordnungen beschreiben. Um auszudrücken, daß die Anordnung, in der o_1 vor o_2 präsentiert wird, bevorzugt wird, werden den Bedingungen, die diese Anordnung beschreiben, höhere Prioritäten zugewiesen.

In CHMIP werden Ablaufpläne unter Berücksichtigung der Ressourcensituation zusammengestellt. Der Ablaufplaner versucht Start- und Endzeitpunkte zu finden, so daß die Präsentation

nicht mehr Ressourcen benötigt als verfügbar sind. Gelingt dies nicht, werden Bedingungen mit niedrigen Prioritäten weggelassen.

2.3.4 MODE

In MODE [BHL91] werden Medien durch eine Menge von Attributen beschrieben. Ein Attribut wird durch einen Namen, einen bevorzugten Wert, einen Wertebereich und eine Priorität beschrieben. Der Wertebereich definiert alle gültigen Attributwerte. Die Priorität definiert die Wichtigkeit des Attributs relativ zu anderen Attributen. Bei der Generierung eines Ablaufplans aus einer Spezifikation werden Werte aus den Wertebereichen der Attribute unter Berücksichtigung der Ressourcensituation gewählt. Werte werden dabei so ausgewählt, daß Attribute mit hohen Prioritätswerten die bevorzugten Werte annehmen. Flexibles zeitliches Verhalten von Medien wird durch Spezifizieren von Wertebereichen für die Präsentationsdauer und Präsentationsrate erreicht.

In MODE kann der Anfang und das Ende eines Mediums sowie jede Dateneinheit eines kontinuierlichen Mediums als Referenzpunkt verwendet werden. Um zeitliche Abhängigkeiten zwischen Medien zu definieren, werden Synchronisationspunkte zwischen Referenzpunkten von Medien spezifiziert. Ein Synchronisationspunkt bewirkt, daß die Präsentation von Medien nicht fortgeführt werden kann, bevor alle beteiligten Medien ihn erreicht haben. Feste Verzögerungen zwischen Medien werden durch Abstraktionen, Timer genannt, modelliert, die durch Synchronisationspunkte mit Referenzpunkten in Beziehung gesetzt werden. Abbildung 2-5 zeigt eine MODE Beispielspezifikation. Die Präsentation in diesem Beispiel beginnt mit einem Video, begleitet von einem Musikstück. Nach dem Ende des Musikstücks und einer Verzögerung von fünf Sekunden wird simultan ein Text und eine Animation präsentiert. Synchronisationspunkte definieren, daß das Video und die Musik gleichzeitig beginnen, aber die Musik erst eine gewisse Zeit nach dem Ende des Videos endet und daß der Text und die Animation gleichzeitig beginnen und enden. Ein Timer realisiert die Verzögerung von fünf Sekunden.

MODE ermöglicht es, für jedes einfache und zusammengesetzte Medium ein oder mehrere Alternativen zu spezifizieren. Eine dieser Alternativen wird präsentiert, wenn es nicht möglich ist, das Medium aufgrund einer Ressourcenknappheit auszuspielen. Für jedes alternative Medium können separate Synchronisationselemente spezifiziert werden.

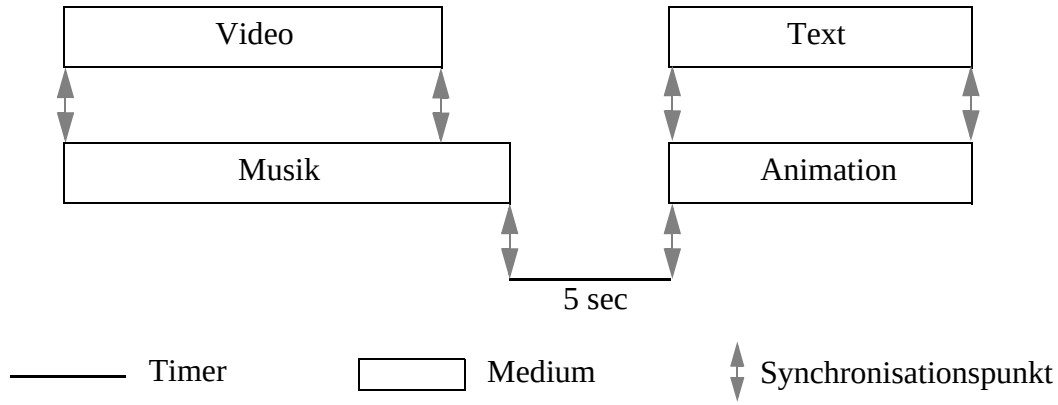


Abb. 2-5 : MODE Spezifikationsbeispiel

2.3.5 Diskussion der Ansätze

Tabelle 2-1 zeigt einen Vergleich der beschriebenen Dokumentenmodelle in Bezug auf ihre Eigenschaften zur Modellierung von Adaptivität.

Alle vorgestellten Dokumentenmodelle integrieren Adaptivität auf Attributebene. Mbuild, Firefly, CHIMP und MODE erlauben die Spezifikation flexibler Präsentationsdauern und flexibler Verzögerungen erlauben. In MODE können außerdem flexible Präsentationsraten spezifiziert werden.

Firefly und MODE kennen Qualitätsmaße für flexible Präsentationsdauern. In MODE können Qualitätsmaße auch für Ratenattribute und flexible Verzögerungen spezifiziert werden. Bei MODE wird jedoch nicht den alternativen Attributwerten, sondern den Attributen an sich eine Priorität zugeordnet, wodurch diese Methode nicht so feine Prioritätszuweisungen erlaubt wie die von Firefly. Demgegenüber ist es in Mbuild und CHIMP nicht möglich die Auswahl von Präsentationsalternativen auf Attributebene zu kontrollieren. Alle möglichen Variationen von Präsentationsdauern und Verzögerungen haben die gleiche Priorität.

Adaptivität auf Objektebene wird nur von CHIMP und MODE unterstützt. MODE ermöglicht die Spezifikation alternativer einfacher und zusammengesetzter Medienobjekte. In CHIMP können weniger wichtige Medienobjekte weggelassen werden und alternative zeitliche Anordnungen von Medienobjekten definiert werden. Das Weglassen von Medienobjekten in CHIMP kann nur sehr schlecht durch den Autor kontrolliert werden, da Medienobjekte automatisch

			Mbuild	Firefly	CHIMP	MODE
Attributebene	Medien-objekte	Flexible Dauern	♦	♦	♦	♦
		Flexible Raten				♦
		Qualitätsmaße		♦		♦
		Auswahlpolitiken				
	Temporale Relationen	Flexible Verzögerungen	♦	♦	♦	♦
		Qualitätsmaße				♦
		Auswahlpolitiken				
Objektebene	Medien-objekte	Präsentationsalternativen				♦
		Weglassen von Objekten			♦	
		Qualitätsmaße			♦	♦
		Auswahlpolitiken				
	Temporale Relationen	Präsentationsalternativen			♦	
		Qualitätsmaße			♦	
		Auswahlpolitiken				

Tabelle 2-1: Vergleich flexibler Multimedia-Dokumentenmodelle

weggelassen werden, wenn nicht alle gleichzeitig präsentiert werden können. Dadurch kann es auch passieren, daß für die Präsentation relevante Medienobjekte nicht ausgespielt werden.

CHIMP erlaubt es, Präsentationsalternativen Prioritäten zuzuweisen. Bei MODE wird die bevorzugte Präsentationsalternative in einer Gruppe vom Präsentationsalternativen markiert. Es wird jedoch keine Priorität zugewiesen, die definiert wie wichtig ein Medienobjekt im Vergleich mit anderen Medienobjekten ist.

Das Fehlen von Adaptivität auf Objektebene in Mbuild und Firefly kommt daher, daß diese Modelle zur Vereinfachung der Spezifikation von Dokumenten und nicht zur Unterstützung adaptiver Präsentationen entwickelt wurden.

2.4 Zusammenfassung und Bewertung

Adaptive Multimedia-Dokumente können alternative Präsentationsmöglichkeiten auf Attribut- und Objektebene integrieren. Es sind Konzepte zur Steuerung und Kontrolle der Auswahl von Präsentationsalternativen erforderlich, da nur so ein maximaler Grad an Flexibilität spezifiziert werden kann. Fehlen diese Mechanismen, können manche Präsentationsalternativen nicht definiert werden, da ein Standardauswahlmechanismus nicht alle Möglichkeiten abdecken wird.

Die existierenden Dokumentenmodelle, die es erlauben Präsentationsalternativen zu definieren, betrachten nur Teilaspekte und integrieren nicht alle der identifizierten Flexibilisierungsmöglichkeiten. Insbesondere was die Auswahlkontrolle von Präsentationsalternativen betrifft, sind die bis jetzt entwickelten Konzepte nicht ausdrucksstark genug.

3 Modellierung adaptiver Multimedia Dokumente

In diesem Kapitel wird das Tiempo-Dokumentenmodell [Wira97, WiRo98] zur Modellierung adaptiver interaktiver Multimedia-Dokumente vorgestellt. Als erstes wird ein Überblick über die aus dem Tiempo-Modell resultierende Tiempo-Dokumentenspracharchitektur gegeben. In Abschnitt 3.2 folgt eine Beschreibung der Abstraktionen zur Modellierung des hierarchischen Aufbaus von adaptiven Multimedia-Dokumenten. Daran schließt sich die Beschreibung des integrierten Interaktionsmodells an. Am Ende des Kapitels wird das Tiempo-Modell mit den in Abschnitt 2.3 vorgestellten adaptiven Dokumentenmodellen verglichen.

3.1 Die Tiempo-Dokumentenspracharchitektur

Multimedia-Dokumente werden in einer Multimedia-Dokumentensprache spezifiziert. Eine Dokumentensprache besteht aus Sprachkonstrukten, die das Dokumentenmodell widerspiegeln. Ein Dokumentenmodell – und damit auch die Dokumentensprache – sollte einfach um neue Medientypen erweiterbar sein, so daß neu entwickelte Medienformate oder spezielle Darstellungen für Mediendaten integriert werden können. Das Tiempo-Modell berücksichtigt diesen Aspekt. Problematisch ist jedoch, daß eine entsprechende Dokumentensprache prinzipiell wie eine Programmiersprache aufgebaut sein muß, wodurch sie für Autoren ohne Programmierkenntnisse ungeeignet ist. Da das Tiempo-Modell aber auch Nicht-Programmierern die Spezifikation von Multimedia-Dokumenten ermöglichen soll, wurde der folgende Ansatz gewählt.

Es wird eine Dokumentenspracharchitektur definiert, die beschreibt wie eine Dokumentensprache aufgebaut sein muß, um dem Tiempo-Modell zu genügen. Anhand dieser Architektur kann je nach Anforderung und Einsatzgebiet eine unterschiedliche konkrete Dokumentensprache abgeleitet werden, die man als **Tiempo-Dialekt** bezeichnet.

Durch diesen Ansatz ist man auch hinsichtlich der Syntax völlig offen. Man kann beispielsweise für den einen Dialekt eine XML-Syntax [BPS98] wählen und für den anderen Dialekt die Syntax einer Programmiersprache wie Java [Flan99], und die Sprachkonstrukte als Erweiterung der Programmiersprache in einer Bibliothek ausliefern.

3.1.1 Das Architekturkonzept

Jede konkrete Tiempo-Dokumentensprache [Wira99a] besteht aus einer Menge von **Dokumentelementtypen**. Dokumente bestehen aus Instanzen der Dokumentelementtypen, die **Dokumentelemente** genannt werden. Für jeden Dokumentelementtyp wird festgelegt, welche Dokumentelemente in welcher Reihenfolge in einem Element dieses Typs enthalten sein können. Auf diese Weise wird die mögliche Struktur von Tiempo-Dokumenten definiert. Die Eigenschaften von Elementen werden durch Attribute näher beschrieben. Bei der Definition eines Elementtyps werden die Attribute mit Standardwerten versehen. Standardwerte kommen in einer Spezifikation zum Tragen, wenn der Autor keinen Wert für ein Attribut definiert hat.

Alle Elementtypen gehen durch Vererbung auseinander hervor. Dies hat den Vorteil, daß Eigenschaften, die viele Elementtypen haben, nur einmal definiert werden müssen. Die Vererbung kann mittels eines Vererbungsbaums dargestellt werden. Abbildung 3-1 zeigt einen solchen Vererbungsbaum. Jeder Elementtyp im Vererbungsbaum erbt die Attribute der Elementtypen von denen er abgeleitet ist, und besitzt damit deren Eigenschaften. Abgeleitete Elementtypen können zusätzliche Attribute besitzen, die weitere Eigenschaften repräsentieren. Mehrfachvererbung ist zugelassen. Elementtypen können abstrakt sein, d.h. von ihnen können keine Instanzen erzeugt werden, und nur der Strukturierung der Vererbungshierarchie dienen. Alle abstrakten Elementtypen sind in Abbildung 3-1 kursiv gesetzt. Der Tiempo-Ansatz beinhaltet somit ähnliche Konzepte wie die Multimedia-Dokumentensprachen MHEG [MHEG94] und HyTime [HyTi92] in denen Sprachelemente ebenfalls durch Vererbung bzw. Ableitung auseinander hervorgehen.

In Abbildung 3-1 bilden die Elementtypen, die über eine durchgezogene Linie mit dem Vererbungsbaum verbunden sind, die Basissprache von Tiempo, welche alle Konzepte des Tiempo-Modells beinhaltet. An die Knoten *Projection*, *InteractionManager*, *SinglemediaProjector*, *MultimediaProjector*, *DiscreteMedia* und *ContinuousMedia* können je nach Tiempo-Dialekt beliebige Elementtypen angehängt werden.

3.1.2 Arten von Elementtypen im Vererbungsbaum

Alle Tiempo-Elementtypen werden vom Basistyp *TSObject* abgeleitet. Der Vererbungsbaum besteht aus zwei Hauptzweigen. Der eine Zweig enthält identifizierbare Elementtypen, die

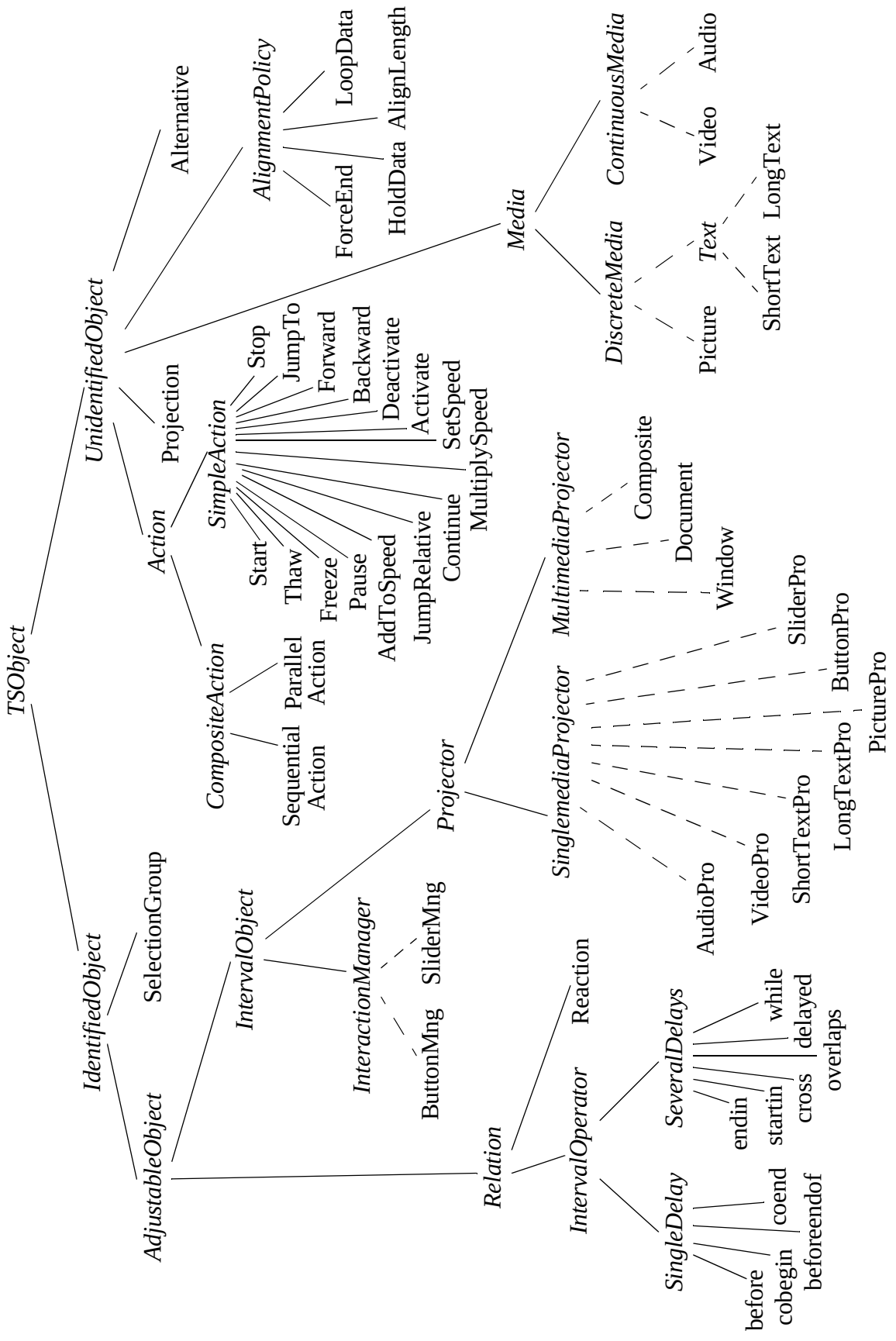


Abb. 3-1: Tiempo-Vererbungsbaum

durch andere Elemente referenziert werden können, der andere Zweig enthält Elementtypen, die nicht referenziert werden können. Der Vererbungsbaum enthält einen Zweig *Media*, der alle Medienelementtypen beinhaltet, die Texte, Bilder, Video-Clips etc. repräsentieren. Instanzen dieser Elementtypen werden **Medienelemente** genannt und beschreiben lediglich die Eigenschaften der repräsentierten Mediendaten. Die konkrete Darstellung von Medien im Rahmen einer Präsentation beschreibt ein dem Medienelement zugeordnetes **Projektorelement**. Die Trennung von Information und deren Darstellung hat den Vorteil, daß dieselbe Information durch Einsatz unterschiedlicher Projektoren verschieden dargestellt werden kann. In Abbildung 3-1 kann ein Medienelement *ShortText* beispielsweise durch ein Projektorelement *ButtonPro* oder *TextPro* dargestellt werden und somit als Überschrift oder Knopfaufschrift verwendet werden. Die Projektoren einfacher Medien befinden sich unterhalb des Knotens *SinglemediaProjector*. Für zusammengesetzte Medien wird nicht zwischen Information und Darstellung unterschieden. Instanzen von Projektorelementtypen für zusammengesetzte Medien werden **Multimedia-Projektoren** genannt und befinden sich unterhalb des Knotens *MultimediaProjector*. Ein einfaches Beispiel für einen solchen Projektor ist ein Fenster (*Window*) das Bilder, Texte und Videos enthalten kann. Zu den Multimedia-Projektoren gehört auch der Elementtyp *Document* der ein Tiempo-Dokument repräsentiert.

Um temporale und interaktive Beziehungen zu definieren, gibt es den Zweig *Relation*. In diesem Zweig sind beispielsweise Elementtypen zu finden, die temporale Operatoren repräsentieren. Der Elementtyp *Reaction* dient zur Beschreibung von Interaktionsbeziehungen. *Reaction*-Elemente beinhalten Instanzen von Elementtypen, die im Zweig *Action* definiert sind. Diese Elementtypen ermöglichen die Spezifikation temporaler Manipulationen von durch Projektorelementen erzeugten Medienpräsentationen. Zur Beschreibung von Präsentationsalternativen auf Dokumentelementebene dient der Elementtyp *SelectionGroup*, der in einem Dokument dann mehrere Elemente des Typs *Alternative* enthält, welche Präsentationsalternativen beschreiben. Elementtypen, die Interaktionsmedien repräsentieren, enthalten Elemente, die sie sensitiv für bestimmte Benutzerinteraktionen machen. Diese werden unterhalb des Knotens *InteractionManager* definiert. Um die Eigenschaften von Projektoren näher zu definieren, werden Elemente vom Typ *Projection* verwendet.

Die vollständige Tiempo-Dokumentenspracharchitektur in UML-Notation [GRJ99] ist in Anhang A.1 zu finden.

3.1.3 Ein Tiempo-Dialekt

Im Rahmen des Tiempo-Projekts wurde die konkrete Tiempo-Dokumentensprache in Abbildung 3-1 durch eine XML Document Type Definition (DTD) definiert. Der XML-Standard [BPS98] hat in kurzer Zeit als Datenbeschreibungssprache in den unterschiedlichsten Bereichen Einzug gehalten. Inzwischen werden unterschiedliche Werkzeuge für XML angeboten, wie beispielsweise XML-Parser. Zur Zeit werden sogar Datenbanken entwickelt, die für die Verarbeitung von XML-Dokumenten optimiert sind. XML wurde für Tiempo verwendet, um von diesen Entwicklungen profitieren zu können.

XML ist eine standardisierte Markup-Sprache mit der die Struktur einer Klasse von Dokumenten in Form einer DTD festgelegt wird. Hierbei werden sogenannte Tags definiert, mittels derer Dokumente spezifiziert werden können. In der Tiempo-DTD wurde für jeden nicht-abstrakten Elementtyp des Dialekts ein entsprechender XML-Elementtyp definiert. Da DTD keine Konzepte wie Vererbung kennt, mußten für jeden XML-Elementtyp immer wieder alle Attribute spezifiziert werden. Um erkennen zu können, von welchen Tiempo-Elementtypen ein entsprechender XML-Elementtyp abgeleitet ist, werden diese in einem besonderen Attribut jedes XML-Elements unveränderbar spezifiziert.

Die DTD des Tiempo-Dialekts ist in Anhang A.2 zu finden. Die wichtigsten Konstrukte, insbesondere im Hinblick auf die Modellierung von Adaptivität, werden in den folgenden Abschnitten vorgestellt.

3.2 Modellierung hierarchischer Multimedia-Dokumente

Um die temporalen Eigenschaften diskreter, kontinuierlicher und zusammengesetzter Medien geeignet beschreiben zu können, wurde in Tiempo das Konzept der Datenräume eingeführt. Durch Projektionen können aus existierenden Datenräumen neue Datenräume mit veränderter Charakteristik erzeugt werden.

3.2.1 Datenräume

Die zeitlichen und räumlichen Aspekte eines Mediums werden durch einen Datenraum repräsentiert. Ein **Datenraum** ist ein vier-dimensionales Koordinatensystem, in dem die Datenein-

heiten des Mediums angeordnet sind. Die Größe eines Datenraums wird durch endliche Koordinatenachsen beschrieben. Eine vertikale y-, horizontale x- und in die Tiefe gerichtete z-Achse erlauben die räumliche Anordnung und eine Zeitachse t die zeitliche Anordnung der Dateneinheiten. Bei Medien, die eine oder mehrere dieser Dimensionen nicht besitzen, haben die entsprechenden Koordinatenachsen die Länge Null. Auf diese Weise können Audios, die keine räumliche Dimension haben, und diskreten Medien, die keine zeitliche Dimension haben, modelliert werden. Wird ein Datenraum entlang der t-Achse mit einer definierten Geschwindigkeit durchlaufen, traversiert man eine Folge von zu präsentierenden Dateneinheiten. Da die Länge der t-Achse eines Mediums dessen Präsentationsdauer beschreibt, gehört das Tiempo-Modell zur Klasse der Intervall-basierten Zeitmodelle. Datenräume können mit Events in der Dokumentensprache HyTime [HyTi92] verglichen werden.

3.2.1.1 Modellierung einfacher Medien

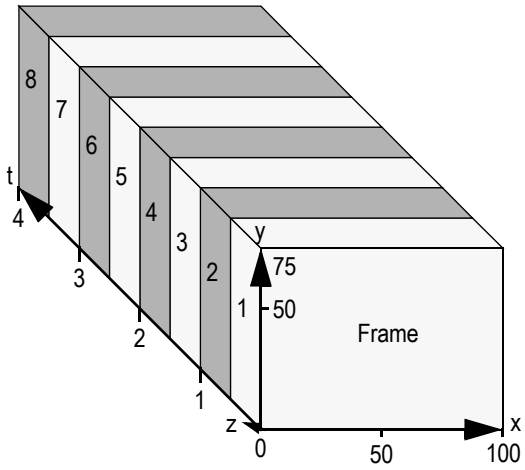
Einfache Medien werden durch **Medienelemente** repräsentiert, deren Attribute je nach Typ des Mediums einen geeigneten Datenraum beschreiben. Die Datenräume von einfachen Medien nennt man **einfache Datenräume**. Hat ein einfaches Medium eine räumliche Dimension, sind die x- und y-Achsen so lang, wie das Medium breit und hoch ist. Da wir keine dreidimensionalen Medien betrachten, hat die z-Achse die Länge Null. Die t-Achse kontinuierlicher Medien ist so lang wie die natürliche Präsentationslänge des Mediums, und die Dateneinheiten des Mediums sind entsprechend seiner natürlichen Rate auf der Zeitachse angeordnet. Wurde beispielsweise ein Video-Clip von vier Sekunden mit zwei Bildern pro Sekunde aufgezeichnet, würde der zugehörige Datenraum wie in Abbildung 3-2a aussehen und durch ein Medienelement vom Typ *Video* beschrieben werden:

```
<!ELEMENT  Video  EMPTY >

<!ATTLIST  Video
        Tiempo      NMTOKEN          #FIXED  "ContinuousMedia"
        Width       CDATA            #REQUIRED
        Height      CDATA            #REQUIRED
        Length      CDATA            #REQUIRED
        Rate        CDATA            #REQUIRED
        Coding      (MPEG | MJPEG)   #REQUIRED
        Data        CDATA            #REQUIRED
>
```

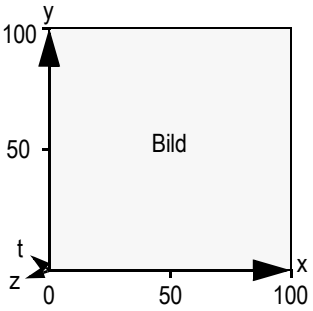
a) Kontinuierliches Medium

```
<Video    Width="100" Height="75"
          Length="4" Rate="2"
          Coding="MPG"
          Data="www.server.com/
                video.mpg"
/>
```



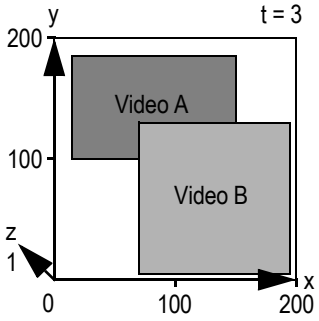
b) Diskretes Medium

```
<Picture  Width="100" Height="100"
          Coding="JPG"
          Data="www.server.com/
                bild.jpg"
/>
```



c) Zusammengesetztes Medium

```
<Window  Id="C" Width="200" Height="200" >
        <VideoPro  Id="B" XPos="80" YPos="130" ZPos="0" >
            <Video  Width="130" Height="130" Length="6" Rate="1" Coding= . . .  />
        </VideoPro>
        <VideoPro  Id="A" XPos="10" YPos="100" ZPos="1" >
            <Video  Width="140" Height="90" Length="4" Rate="2" Coding= . . . />
        </VideoPro>
        <Delayed  From="A" To="B" Delay1="3" Delay2="5" />
        <Coend  From="B" To="C" Delay1="2" />
</Window>
```



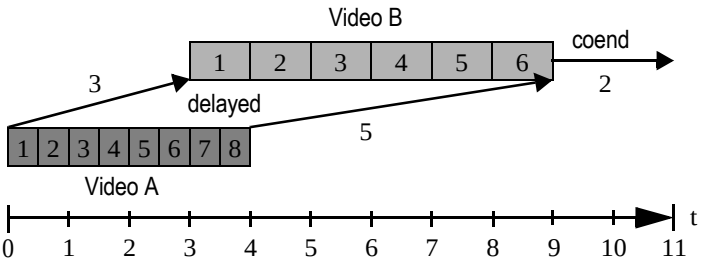


Abb. 3-2: Datenräume der Medienklassen

Die Attribute *Width*, *Height*, *Length* und *Rate* definieren den Datenraum, der die mit dem Verfahren *Coding* kodierten Daten *Data* enthält.

Bei diskreten Medien hat die t-Achse die Länge Null. Auf der t-Achse ist dem Zeitpunkt Null das Datum des Mediums zugeordnet, wie in Abbildung 3-2b dargestellt. Der Datenraum eines diskreten Mediums ist somit durch die Angabe der Länge der x- und y-Achsen vollständig definiert. Das diskrete Medium in Abbildung 3-2b ist vom Medienelementtyp *Picture*:

```
<!ELEMENT  Picture  EMPTY >

<!ATTLIST  Picture
        Tiempo      NMTOKEN             #FIXED  "DiscreteMedia"
        Width       CDATA               #REQUIRED
        Height      CDATA               #REQUIRED
        Coding      (GIF | JPEG | XPM)  #REQUIRED
        Data        CDATA               #REQUIRED
>
```

Die Attribute *Width* und *Height* definieren den Datenraum, der die mit dem Verfahren *Coding* kodierten Daten *Data* enthält.

3.2.1.2 Modellierung zusammengesetzter Medien

Die räumlichen und zeitlichen Eigenschaften zusammengesetzter Medien werden durch **zusammengesetzte Datenräume** repräsentiert. In einem zusammengesetzten Datenraum werden die Datenräume, der im zusammengesetzten Medium enthaltenen Medien, entsprechend dem gewünschten räumlichen und zeitlich Layout angeordnet. In zusammengesetzten Datenräumen kann es räumliche und zeitliche Bereiche geben, denen keine Dateneinheiten zugeordnet sind.

In Abbildung 3-2c ist auf der rechten Seite die t-Achse eines zusammengesetzten Datenraums zu sehen, in dem sich zwei kontinuierliche Medien zwischen Zeiteinheit drei und vier überlappen, und auf der linken Seite die drei räumlichen Dimensionen zum Zeitpunkt drei auf der t-Achse. Zu diesem Zeitpunkt werden beide enthaltenen Medien präsentiert, wobei auf der z-Achse Video *B* vor Video *A* liegt.

Das Tiempo-Modell verwendet unterschiedliche Ansätze zur Spezifikation des räumlichen und zeitlichen Layouts zusammengesetzter Datenräume. Zur räumlichen Anordnung werden die Datenräume enthaltener Medien absolut auf den räumlichen Koordinatenachsen des zusammengesetzten Datenraums positioniert. Dazu ist jedes Medienelement durch ein **einfaches Projektorelement** gekapselt, das die Positionierung definiert. In Abbildung 3-2c wird beispielsweise die Position des Datenraums von Video *B* im zusammengesetzten Datenraum von Window *C* durch die Attribute *XPos*, *YPos* und *ZPos* des Projektorelements *B* festgelegt, das wie folgt definiert ist:

```
<!ELEMENT  VideoPro  (Video) >

<!ATTLIST  VideoPro
        Tiempo      NMTOKEN     #FIXED  "SinglemediaProjector"
        Id          ID          #REQUIRED
        XPos        CDATA       "0"
        YPos        CDATA       "0"
        ZPos        CDATA       "0"
>
```

Da mit einem Projektorelementtyp eine bestimmte Darstellung von Mediendaten assoziiert wird, kann jeder Projektorelementtyp nur bestimmte Medienelementtypen enthalten. Das Attribut *Id* dient zur Identifikation eines Projektorelements.

Um die Integration von zeitlicher Flexibilität und von Benutzerinteraktionen einfach zu gestalten, werden enthaltene Datenräume auf der t-Achse eines zusammengesetzten Datenraums durch **Intervalloperatoren** relativ zueinander positioniert. Wie in [Alle83] gezeigt wurde, gibt es zwischen zwei kontinuierlichen Medien eine Vielzahl zeitlicher Anordnungsmöglichkeiten. In [WaRo94] wurden die relevanten Anordnungen herausgefiltert und zehn sogenannte Intervalloperatoren entwickelt, mittels derer diese Anordnungen beschrieben werden können. Abbildung 3-3 zeigt die Semantik der Intervalloperatoren.

Intervalloperatoren können auf Datenräume angewendet werden, weil es bei deren zeitlicher Anordnung nur auf die Länge der t-Achse ankommt, die als Zeitintervall aufgefaßt werden kann. Spezifiziert man beispielsweise einen *delayed*-Operator wie in Abbildung 3-2c, wird Video *B* drei Zeiteinheiten nach Video *A* im umgebenden Datenraum beginnen und Video *B* fünf Zeiteinheiten nach Video *A* enden. Der *delayed*-Operator ist hierbei definiert als:

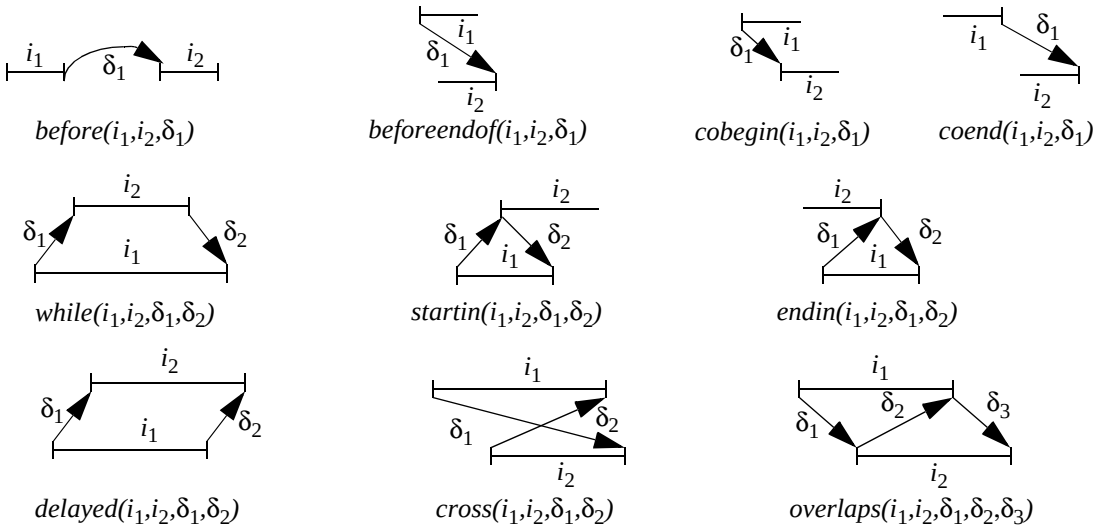


Abb. 3-3: Intervalloperatoren

```
<!ELEMENT  Delayed  EMPTY >

<!ATTLIST  Delayed
        Tiempo      NMTOKEN     #FIXED  "IntervalOperator"
        Id          ID          #REQUIRED
        From        CDATA       #REQUIRED
        To          CDATA       #REQUIRED
        Delay1      CDATA       "[0,?]"
        Delay2      CDATA       "[0,?]"
>
```

Intervalloperatoren können zwischen Elementen spezifiziert werden, die sich in unterschiedlichen Datenräumen befinden. Dadurch entstehen sogenannte **zeitliche Querbeziehungen**, die es dem Autor erlauben, ein Dokument beliebig durch zusammengesetzte Medien zu strukturieren und dennoch alle gewünschten Zeitbeziehungen definieren zu können.

Intervalloperatoren haben erhebliche Vorteile gegenüber Operatoren, die generische Anfang-Ende-Beziehungen zwischen Medien beschreiben. Ein Intervalloperator definiert die vollständige Beziehung zwischen zwei Medien. Dadurch wird die Anzahl der in einem Dokument benötigten Operatoren minimiert, was einerseits den Spezifikationsaufwand reduziert und andererseits die Spezifikationen überschaubarer macht. Die Intervalloperatoren *while*, *delayed*, *startin*, *endin*, *cross*, *overlaps* sind überspezifiziert, wenn alle Verzögerungsparameter δ_i spezifiziert werden. Der Autor kann sich also aussuchen, welchen Parameter er spezifizieren möchte. Die

anderen Parameter behalten ihren Standardwert [0,?], der dem Werteintervall $[0, \infty]$ entspricht, da ihr Wert vom Dokumentensystem festgelegt werden kann. Des weiteren haben die Intervalloperatoren sprechende Namen, und sind daher sehr intuitiv anwendbar.

Um Autoren Rechenarbeit abzunehmen, wird die Länge der t- und z-Achse eines zusammengesetzten Datenraums automatisch bestimmt. Diese Achsen haben immer die minimal mögliche Länge. Das ist die Länge, bei der die enthaltenen Datenräume entsprechend der spezifizierten Anordnung genau hineinpassen. Die Intervalloperatoren *cobegin*, *coend* und *while* können zur Festlegung des Abstands eines enthaltenen Datenraums zum Anfang oder Ende des umgebenden Datenraums verwendet werden. In Abbildung 3-2c wird beispielsweise durch einen *coend*-Operator festgelegt, daß der umgebende Datenraum zwei Zeiteinheiten nach dem Ende von Video *B* endet. Da zwischen Video *A* und dem umgebenden Datenraum kein Intervalloperator spezifiziert wurde, beginnt der zusammengesetzte Datenraum direkt mit Video *A*. In der x- und y-Dimension legt der Autor die Länge der Achsen fest.

Im Tiempo-Modell werden zusammengesetzte Medien durch Multimedia-Projektorelemente repräsentiert. Das zusammengesetzte Medium in Abbildung 3-2c, welches ein einfaches Fenster repräsentiert, ist beispielsweise durch folgenden Multimedia-Projektortyp definiert:

```
<!ELEMENT  Window  (VideoPro | PicturePro | LongTextPro | ShortTextPro | AudioPro |
                    Composite | ButtonPro | SliderPro)+,
                    (Before | Cobegin | Coend | Beforeendof | While | Delayed |
                    Startin | Endin | Cross | Overlaps)* >

<!ATTLIST  Window
      Tiempo        NMTOKEN     #FIXED  "MultimediaProjector"
      Id            ID          #REQUIRED
      Width         CDATA       "300"
      Height        CDATA       "300"
      XPos          CDATA       "0"
      YPos          CDATA       "0"
      ZPos          CDATA       "0"
      Background    CDATA       "grey"
>
```

Ein Multimedia-Projektor besitzt die Eigenschaften eines Medienelements und eines Projektorelements. In einem Multimedia-Projektorelement werden die Projektorelemente und Intervalloperatoren spezifiziert, die seinen Datenraum definieren. Die Länge der x- und y-Achse des

Datenraums kann durch die Attribute *Width* und *Height* vom Autor definiert werden. Ein Multimedia-Projektorelement hat zudem die Attribute *XPos*, *YPos* und *ZPos*, um den auf diese Weise definierten Datenraum in einem anderen zusammengesetzten Datenraum anzuordnen. Dies ermöglicht es, Dokumente hierarchisch zu strukturieren.

Welche Elementtypen in einem Multimedia-Projektorelement enthalten sein können, hängt von seiner Semantik ab. Der im Rahmen dieser Arbeit verwendete Tiempo-Dialekt kennt beispielsweise drei Multimedia-Projektortypen. Der Typ *Document* repräsentiert ein Multimedia-Dokument, das aus mehreren Fenstern besteht. Daher kann ein *Document*-Element nur Elemente vom Typ *Window* enthalten. Elemente diesen Typs können alle einfachen Projektorelemente enthalten und Multimedia-Projektoren vom Typ *Composite*, die zur weiteren Strukturierung von Spezifikationen dienen. *Composite*-Elemente können alle einfachen Projektorelemente sowie andere *Composite*-Elemente enthalten.

3.2.2 Projektionen

Medien werden häufig an mehreren Stellen einer Präsentation in etwas unterschiedlicher Weise eingesetzt. Um zu vermeiden, daß unterschiedliche Versionen eines Mediums generiert und gespeichert werden müssen, sollte ein Autor die folgenden Spezifikationsmöglichkeiten haben:

- Präsentation eines bestimmten Teils eines Mediums.
- Neukombination von Medienteilen.
- Variation der Präsentationsgeschwindigkeit kontinuierlicher Medien.

Um diesen Anforderungen zu genügen, integriert das Tiempo-Modell **Projektionen** [Wira97]. Eine Projektion generiert aus einem existierenden Datenraum einen neuen Datenraum mit veränderter Charakteristik. Das Konzept der Manipulation von Medien durch Projektion ist nicht neu, es gibt beispielsweise in HyTime [HyTi92] ein ähnliches Konzept. Neu an dem Tiempo-Projektionskonzept ist, daß es eine fixe Anzahl von zeitlichen Projektionstypen definiert, durch deren Kombination alle temporalen Modifikationen eines Mediums beschrieben werden können. Eine temporale Projektion ist wie folgt definiert:

```
<!ELEMENT Projection (AlignLength | HoldData | LoopData | ForceEnd) >

<!ATTLIST Projection
        Tiempo          NMTOKEN     #FIXED  "Projection"
        PresInterval    CDATA       "[0,?]"
        Speed           CDATA       "1.0"
        Rate            CDATA       "1.0"
>
```

Das **Präsentationsintervall** (*PresInterval*) definiert die Länge der t-Achse des neu erzeugten Datenraums. Die **Präsentationsgeschwindigkeit** (*Speed*) beschreibt wie gedehnt bzw. gestaucht die Zeitpunkte des existierenden Datenraums im neuen Datenraum zu liegen kommen. Ein Zeitpunkt identifiziert hierbei die mit ihm korrelierten Dateneinheiten von Medien. Die **Präsentationsrate** (*Rate*) beschreibt wieviele Dateneinheiten pro Zeiteinheit im neuen Datenraum zu liegen kommen. *Speed*- und *Rate*-Werte werden als typenloser Faktor angegeben, mit dem die Präsentationsgeschwindigkeit bzw. -rate im existierenden Datenraum multipliziert wird. Dadurch ist die Spezifikation dieser Werte unabhängig vom Medientyp und der Medienklasse, und es kann beispielsweise die Geschwindigkeit eines Audios genauso wie die Geschwindigkeit eines Videos oder eines zusammengesetzten Mediums verändert werden. Bei der Realisierung einer Projektion während einer Präsentation wird zuerst ein entsprechend dem *Speed*-Wert gestauchter bzw. gedehnter Datenraum erzeugt, in dem alle Dateneinheiten des projizierten Datenraums enthalten sind. Danach werden entsprechend dem *Rate*-Wert nicht benötigte Dateneinheiten entfernt.

In einem *Projection*-Element ist eine **Ausrichtungspolitik** enthalten, die definiert mit welchem Teil des existierenden Datenraums der neue Datenraum zu füllen ist. Tiempo bietet vier Ausrichtungspolitiken an:

- *AlignLength*:

```
<!ELEMENT AlignLength EMPTY>

<!ATTLIST AlignLength
        Tiempo      NMTOKEN     #FIXED  "AlignmentPolicy"
>
```

  Bei dieser Ausrichtungspolitik wird der existierende Datenraum komplett in den neuen Datenraum projiziert. Diese Ausrichtungspolitik impliziert, daß das Produkt aus Präsenta-

tionsintervallänge und Präsentationsgeschwindigkeit gleich der Länge der t-Achse des existierenden Datenraums ist. Es muß daher nur einer der beiden Parameter spezifiziert werden, da der andere dann ebenfalls festliegt. Die Präsentationsrate kann jedoch frei gewählt werden. Abbildung 3-4a zeigt ein Beispiel für die Anwendung der Politik.

- *HoldData:*

```
<!ELEMENT HoldData EMPTY >

<!ATTLIST HoldData
        Tiempo      NMTOKEN             #FIXED  "AlignmentPolicy"
        Instant     CDATA               "0"
        Position    (first | last | center)   "first"
>
```

Bei dieser Ausrichtungspolitik sind Präsentationsintervallänge und Präsentationsgeschwindigkeit unabhängig voneinander. Zeitpunkte aus dem existierenden Datenraum werden entsprechend der spezifizierten Geschwindigkeit im neuen Datenraum angeordnet. Welche Zeitpunkte dies sind, wird durch zwei Parameter spezifiziert. Der Parameter *Instant* referenziert einen Zeitpunkt im existierenden Datenraum und der Parameter *Position* definiert, wo dieser Zeitpunkt im neuen Datenraum zu liegen kommt. Hat *Position* den Wert:

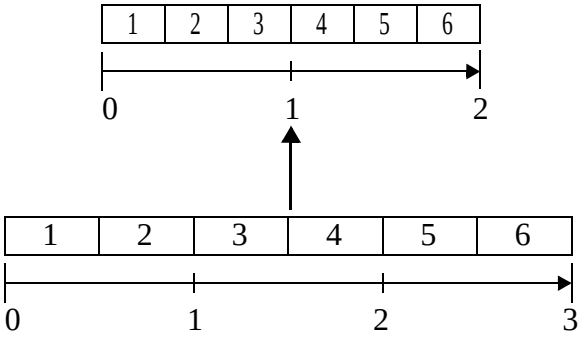
- *first,* wird der Zeitpunkt *Instant* auf den ersten Zeitpunkt des neuen Datenraums projiziert.

- *center,* wird der Zeitpunkt *Instant* in die Mitte des neuen Datenraum projiziert.

- *last,* wird der Zeitpunkt *Instant* auf den letzten Zeitpunkt des neuen Datenraum projiziert.

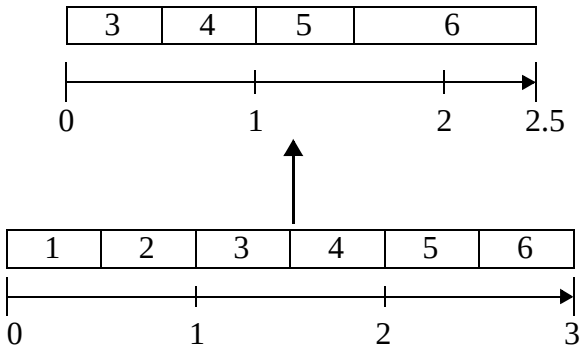
Die anderen Zeitpunkte des existierenden Datenraums werden im neuen Datenraum so angeordnet, daß sie zu *Instant* wie im existierenden Datenraum zu liegen kommen, jedoch nicht aus dem neuen Datenraum herausragen. Entsteht aufgrund dieser Anordnung eine Lücke am Anfang oder Ende des neuen Datenraums, wird sie durch den ersten bzw. letzten projizierten Zeitpunkt gefüllt. Abbildung 3-4b zeigt ein Beispiel für die Anwendung dieser Politik, bei der am Ende des neuen Datenraums eine Lücke entsteht, welche durch den letzten ausgespielten Zeitpunkt gefüllt wird.

a)



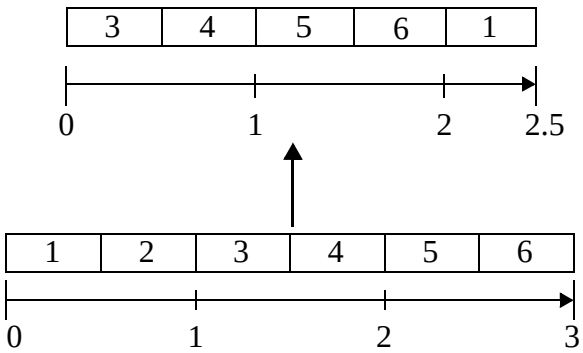

```
<Projection  PresInterval="2"
             Speed="1.5" Rate="1.0" >
      <AlignLength/>
</Projection>
```

b)



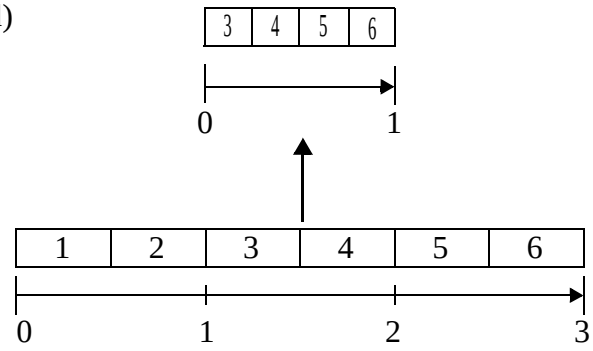

```
<Projection  PresInterval="2.5"
             Speed="1.0" Rate="1.0" >
      <HoldData  Instant="1"
                 Position="first" />
</Projection>
```

c)




```
<Projection  PresInterval="2.5"
             Speed="1.0" Rate="1.0" >
      <LoopData  Instant="1"
                 Position="first" />
</Projection>
```

d)




```
<Projection  PresInterval="1"
             Speed="2.0" Rate="1.0" >
      <ForceEnd  Instant="1"
                 Position="first" />
</Projection>
```

Abb. 3-4: Projektionstypen

- *LoopData:*

```
<!ELEMENT LoopData EMPTY >

<!ATTLIST LoopData
        Tiempo      NMTOKEN             #FIXED  "AlignmentPolicy"
        Instant     CDATA               "0"
        Position    (first | last | center)     "first"
>
```

  Diese Ausrichtungspolitik hat die gleiche Semantik wie *HoldData*, bis auf die Art und Weise wie Lücken gefüllt werden. Im Unterschied zu *HoldData* werden hier Lücken fortlaufend durch Zeitpunkte des projizierten Datenraums gefüllt, und zwar so als wäre der projizierte Datenraum eine unendliche Verkettung von sich selbst. Abbildung 3-4c zeigt ein Anwendungsbeispiel für diese Politik.

- *ForceEnd:*

```
<!ELEMENT ForceEnd EMPTY >

<!ATTLIST ForceEnd
        Tiempo      NMTOKEN             #FIXED  "AlignmentPolicy"
        Instant     CDATA               "0"
        Position    (first | last | center)     "first"
>
```

  Bei dieser Ausrichtungspolitik sind Präsentationsintervallänge und Präsentationsgeschwindigkeit unabhängig. Zeitpunkte des gegebenen Datenraums werden mit der spezifizierten Geschwindigkeit in den neuen Datenraum projiziert. Welche Zeitpunkte dies sind, wird durch die Parameter *Instant* und *Position* spezifiziert. Bei dieser Politik dürfen jedoch keine zusätzlichen Lücken im neuen Datenraum entstehen. Abbildung 3-4d zeigt ein Anwendungsbeispiel für diese Politik.

Für diskrete Medien wird die Ausrichtungspolitik *HoldData* verwendet, mit den Parametern *Instant = "0"* und *Position = "first"*. Zudem wird die Präsentationsgeschwindigkeit auf null und die Präsentationsrate auf eins gesetzt. Dies bewirkt, daß die t-Achse des neuen Datenraums die Länge des Präsentationsintervalls hat und jedem Zeitpunkt das Datum des diskreten Mediums zugeordnet ist. Für kontinuierliche und zusammengesetzte Medien kann jede Ausrichtungspolitik gewählt werden.

Projection-Elemente werden innerhalb von Projektorelementen spezifiziert. Ein Projektor kombiniert eine Projektion und einen existierenden einfachen bzw. zusammengesetzten Datenraum und positioniert den neu entstehenden Datenraum in einem umgebenden Datenraum. Ein Videoprojektortyp, der eine Projektion durchführen kann, ist beispielsweise wie folgt definiert:

```
<!ELEMENT  VideoPro  (Projection, Video) >

<!ATTLIST  VideoPro
        Tiempo      NMTOKEN     #FIXED  "SinglemediaProjector"
        Id          ID          #REQUIRED
        XPos        CDATA       "0"
        YPos        CDATA       "0"
        ZPos        CDATA       "0"
>
```

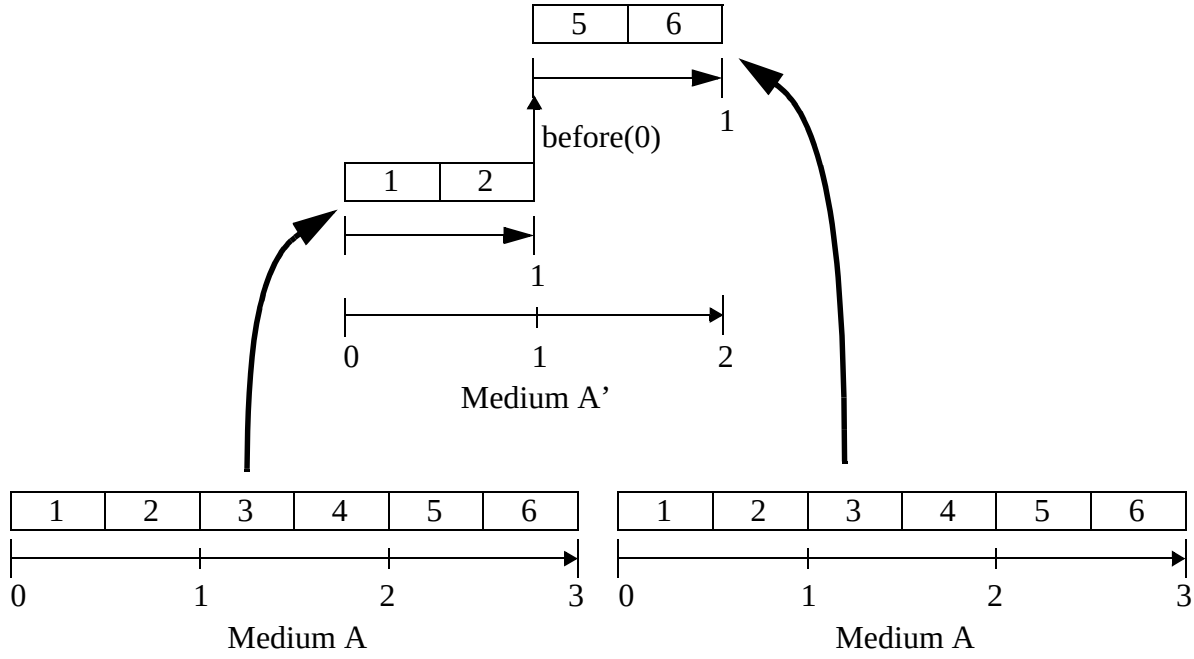
Entsprechend der Tiempo-Dokumentenspracharchitektur enthält jeder Projektor eine Projektion. Wird keine Manipulation gewünscht, wird eine identische Abbildung eines Datenraums wie folgt spezifiziert:

```
<Projection  PresInterval="[0,?]" Speed="1.0" Rate="1.0" >
        <AlignLength/>
</Projection>
```

Durch die Kombination mehrerer Projektionen können Medienteile neu zusammengestellt werden. Abbildung 3-5 zeigt beispielsweise, wie mittels Projektionen der mittlere Teil eines Mediums *A* weggelassen werden kann, so daß ein Medium *A'* entsteht, welches lediglich aus dem Ende und dem Anfang des Mediums *A* besteht. Projektion eröffnen dem Autor eines Multimedia-Dokuments somit vielfältige Möglichkeiten Medien bzw. Teile von Medien entsprechend seinen Vorstellungen zu verwenden. Ist es notwendig auch räumliche Manipulationen von Datenräumen vorzunehmen, können entsprechende Projektionen vom *Projection*-Elementtyp abgeleitet werden.

3.3 Interaktionen

Insbesondere bei Multimedia-Lerndokumenten soll die Präsentation nicht wie ein Film ablaufen. Der Nutzer sollte die Möglichkeit haben eine Präsentation zu beeinflussen, beispielsweise indem er einen Teil noch einmal wiederholen kann, wenn er etwas nicht verstanden hat. Inter-




```
<Composite  Id="A" >
      <Projection  PresInterval="[0,?]" Speed="1.0" Rate="1.0" >
          <AlignLength/>
      </Projection>
      <VideoPro  Id="PA1" >
          <Projection  PresInterval="1" Speed="1.0" Rate="1.0" >
             <ForceEnd  Instant="0" Position="first" />
          </Projection>
          <Video  Length="3" Rate="2" Data="www.server.com/videoA.mpg" />
      </VideoPro>
      <VideoPro  Id="PA2" >
          <Projection  PresInterval="1" Speed="1.0" Rate="1.0" >
             <ForceEnd  Instant="2" Position="first" />
          </Projection>
          <Video Length="3" Rate="2" Data="www.server.com/videoA.mpg" />
      </VideoPro>
      <Before  From="PA1" To="PA2" Delay1="0" />
</Composite>
```

Abb. 3-5: Neukombination von Medienteilen durch Projektionen

aktionen sind daher ein wichtiges Element von Multimedia-Dokumentenmodellen [WRW95, WWR95]. In dieser Abhandlung werden die zeitlichen Interaktionen einer genaueren Betrachtung unterzogen.

Bei zeitlichen Interaktionen interagiert der Nutzer meist nicht direkt mit den zu manipulierenden Medien, sondern mit Interaktionsmedien wie Knöpfen oder Schiebereglern, welche dann die Präsentation der zu manipulierenden Medien verändern. Um eine solche Interaktion in einem Dokument beschreiben zu können, müssen die Interaktionsmöglichkeiten der Interaktionsmedien und die durch sie auslösbaren Interaktionseffekte modelliert werden.

3.3.1 Interaktionsmöglichkeiten

Interaktionsmöglichkeiten, die Interaktionsmedien bieten, werden durch **Interaktionsmanager** beschrieben. Dasselbe Interaktionsmedium kann verschiedene Arten von Benutzerinteraktionen akzeptieren. Beispielsweise kann der Klick mit der rechten Maustaste etwas anderes bewirken als der Klick mit der linken Maustaste. Einem Interaktionsmedium wird für jede mögliche Art von Benutzerinteraktion ein separater Interaktionsmanager zugeordnet. Interaktionsmanagerelemente werden in Projektorelementen, die Interaktionsmedien repräsentieren, spezifiziert. Projektorelemente die Interaktionsmanager enthalten, heißen **Interaktionsprojektoren**. Der Interaktionsmanager eines Knopfs ist beispielsweise definiert als:

```
<!ELEMENT ButtonIntMng EMPTY >

<!ATTLIST ButtonIntMng
        Tiempo      NMTOKEN             #FIXED  "InteractionManager"
        Id          ID                  #REQUIRED
        IntInterval CDATA               "[0,?]"
        StartDelay  CDATA               "[0,?]"
        EndDelay    CDATA               "[0,?]"
        State       (pressed | released)  "released"
>
```

Jeder Interaktionsmanagertyp hat die Attribute *IntInterval*, *StartDelay*, *EndDelay* und *State*. Das Attribute *State* repräsentiert den **Interaktionszustand** des Interaktionsprojektors. Tritt eine Benutzerinteraktion auf, ändert sich sein Wert. Laut dem *ButtonIntMng* kann ein Knopf zwei Interaktionszustände annehmen, entweder ist er gedrückt oder nicht gedrückt. Bei einem Schieberegler kann jede eingestellte Position ein separater Interaktionszustand sein. Aufgrund der unterschiedlichen Interaktionszustände, die ein Interaktionsmedium einnehmen kann, gibt es verschiedene Interaktionsmanagerelementtypen, die alle vom Elementtyp *InteractionManager*

abgeleitet werden, der die Attribute *IntInterval*, *StartDelay* und *EndDelay* definiert. Die konkrete Ausprägung des *State*-Attributes wird durch die abgeleiteten Elementtypen definiert.

Das *IntInterval*-Attribut beschreibt die Dauer der sensitiven Phase des Interaktionsmanagers, das sogenannte **Interaktionsintervall**. In der sensitiven Phase werden Benutzeraktionen auf dem Interaktionselement verarbeitet. Das Interaktionsintervall liegt logisch im Datenraum in den der Interaktionsprojektor, in welchem der Interaktionsmanager enthalten ist, hineinprojiziert. Das Attribut *StartDelay* definiert die Verzögerung zwischen dem Start der Präsentation des Interaktionsmediums und der sensitiven Phase des Interaktionsmanagers. Entsprechend beschreibt das Attribut *EndDelay* die Verzögerung zwischen dem Ende der aktiven Phase des Interaktionsmanagers und dem Ende der Präsentation des Interaktionsmediums.

Um die sensitive Phase eines Interaktionsmanagers zu spezifizieren, gibt es zwei Möglichkeiten. Die Länge des Interaktionsintervalls kann direkt durch Angabe der Attribute *IntInterval*, *StartDelay* und *EndDelay* definiert werden oder indirekt durch die Definition von Abhängigkeiten zu anderen Intervallen spezifiziert werden. Bei der indirekten Spezifikation wird das Interaktionsintervall mittels Intervalloperatoren zu anderen Intervallen in Beziehung gesetzt. Das Interaktionsintervall wird hierbei über den Interaktionsmanager identifiziert. Um beispielsweise zu beschreiben, daß ein Knopf erst zehn Zeiteinheiten nach Beginn einer Video-Präsentation sensitiv sein soll, aber zur gleichen Zeit wie das Video präsentiert werden soll, wird das Präsentationsintervall des Videoelements durch einen *cobegin*-Operator mit dem Interaktionsintervall des Knopfs verbunden und der Verzögerungsparameter des Intervalloperators auf zehn gesetzt sowie ein *while*-Operator zwischen den Video- und Knopf-Projektoren spezifiziert:

```
<Window ...>
   <VideoPro  Id="PV" ...>
      <Projection  PresInterval="600" Speed="1.0" Rate="1.0" >
         <AlignLength/>
      </Projection>
      <Video ... />
   </VideoPro>
   <ButtonPro  Id="PB" ...>
      <Projection  PresInterval="[0,?]" Speed="0" Rate="1.0" >
         <HoldData  Instant="0" Position="first" />
      </Projection>
      <ButtonIntMng  Id="IB" IntInterval="[0,?]" StartDelay="[0,?]"
                     EndDelay="[0,?]" State="released" />
```

```
        <ShortText  Data="Video Stoppen" />
    </ButtonPro>
    <While  Id="W" From="PV" To="PB" Delay1="0" Delay2="0" />
    <Cobegin  Id ="C" From="PV" To="IB" Delay1="10" />
</Window>
```

3.3.2 Interaktionseffekte

Interaktionseffekte werden im Tiempo-Modell durch **Reaktionsoperatoren** spezifiziert, die wie folgt definiert sind:

```
<!ELEMENT  Reaction (SequentialAction | ParallelAction | Activate | Deactivate |
                     Start | Stop | JumpTo | JumpRelative | Pause | Continue |
                     Freeze | Thaw | AddToSpeed | MultiplySpeed | SetSpeed |
                     Forward | Backward ) >

<!ATTLIST  Reaction
      Tiempo          NMTOKEN     #FIXED  "Reaction"
      Id              ID          #REQUIRED
      IntManager      IDREF       #REQUIRED
      PreCondition    CDATA       #IMPLIED
      PostCondition   CDATA       #IMPLIED
>
```

Ein *Reaction*-Operator referenziert im Attribut *IntManager* den Interaktionsmanager, der den Interaktionseffekt auslöst, und enthält **Aktionen**, die den Interaktionseffekt beschreiben. *Reaction*-Elemente werden in Multimedia-Projektoren spezifiziert.

Um das Auftreten eines Interaktionseffekts einzuschränken, enthält das *Reaction*-Element ein Attribut *PreCondition*, das definiert, welchen Interaktionszustand der referenzierte Interaktionsmanager vor Auftreten der Benutzerinteraktion besitzen soll, und ein Attribut *PostCondition*, das definiert, welchen Interaktionszustand er nach Auftreten der Interaktion besitzen soll. Die Aktionen im *Reaction*-Operator werden genau dann ausgeführt, wenn ein Zustandswechsel entsprechend der Attribute *PreCondition* und *PostCondition* auftritt. Ist für eines oder beide Attribute kein Wert spezifiziert, erfüllt jeder Interaktionszustand die Vor bzw. Nachbedingung. Wird beispielsweise nur die Nachbedingung definiert, wird bei jeder Zustandsänderung des Interaktionsmanagers, die zu dieser Nachbedingung führt, die Aktion durchgeführt. Eine Dokumentspezifikation könnte folgenden Zeilen enthalten, um zu definieren, daß bei Drücken des Knopfs *B* das Video *V* starten soll:

```
<Window ...>
        <ButtonPro Id="B" ...>
                <Projection  PresInterval="[0,?]" Speed="0" Rate="1.0" >
                        <HoldData  Instant="0" Position="first" />
                </Projection>
                <ButtonIntMng  Id="BM" IntInterval="[0,?]" StartDelay="0" EndDelay="0"
                                State="released" />
                <ShortText  Data="Video Starten" />
        </ButtonPro>
        <VideoPro Id="V" ... >
                <Projection  .../>
                <Video .../>
        </VideoPro>
        <Reaction IntManager="BM" PreCondition="pressed" PostCondition="released">
                <Start  Element="V" />
        </Reaction>
        . . .
</Window>
```

Jeder Interaktionseffekt kann als Funktion

$$C_{i+1} = F(C_i, M) \tag{3-1}$$

beschrieben werden, die den aktuellen Präsentationskontext C_i in einen neuen Präsentationskontext C_{i+1} überführt. M ist hierbei eine Menge von Parametern, welche die durch F bewirkte Modifikation näher definiert. Der Präsentationskontext C_i ist definiert als Menge der Präsentationszustände aller an der Präsentation beteiligten Elemente. Der Präsentationszustand eines Projektorelements i kann als Tripel (z_i, v_i, t_i) beschrieben werden. Hierbei ist v_i die Geschwindigkeit mit der die Zeit im Datenraum des Mediums vergeht, t_i der Zeitpunkt im Datenraum, der gerade präsentiert wird, und z_i der Elementzustand in dem sich der Projektor befindet. Der Präsentationszustand aller anderen Elementarten ist alleine durch deren Elementzustand z_i definiert. Welche Elementzustände für ein Element existieren, hängt von den möglichen Interaktionseffekten ab.

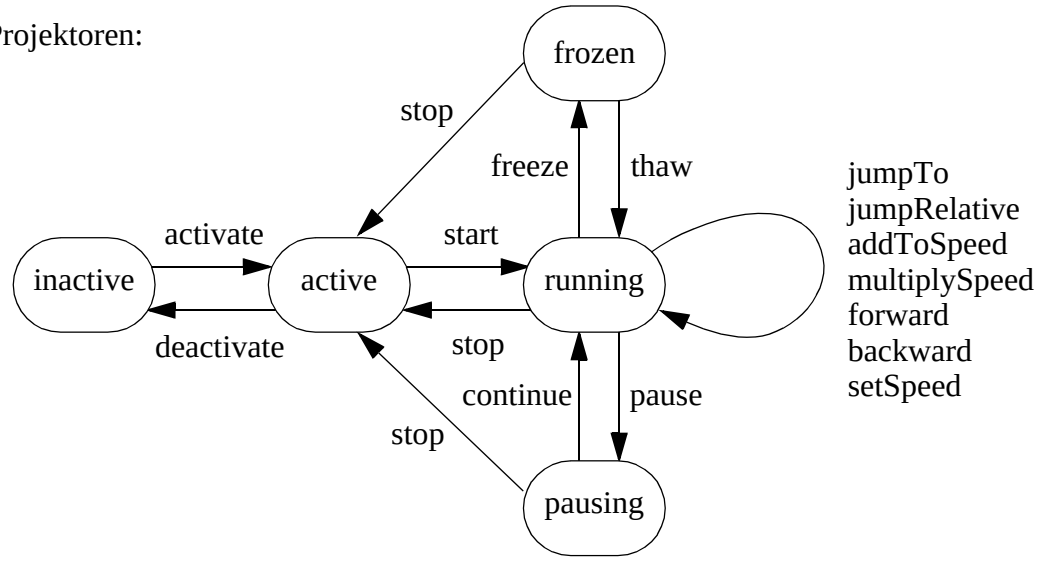
Zur Modellierung von Interaktionseffekten bietet das Tiempo-Modell **elementare Aktionen** an. Die erste Spalte von Tabelle 3-1 zeigt die verfügbaren Aktionen mit den entsprechenden Parametern. Die zweite Spalte zeigt in welchem Elementzustand sich das Element befinden muß, damit die Aktion ausgeführt werden kann. Die letzten drei Spalten zeigen die durch die Aktionen ausgelöste Veränderung des Präsentationszustandes eines Elements.

Elementare Aktion	Elementzustand		Geschwindigkeit	Medienzeit
	z_{alt}	z_{neu}	v_{neu}	t_{neu}
start(Id)	active	running	$v_{spezifiziert}$	0 | Instant
stop(Id)	ru., fr., pa.	active	-	-
jumpTo(Id, $t_{absolut}$)	running	running	v_{alt}	$t_{absolut}$
jumpRelative(Id, t_{delta})	running	running	v_{alt}	$t_{alt} + t_{delta}$
pause(Id)	running	pausing	0	t_{alt}
continue(Id)	pausing	running	v_{vor_pause}	t_{alt}
freeze(Id)	running	frozen	0	t_{alt}
thaw(Id)	frozen	running	v_{vor_thaw}	$t_{aktuell}$
addToSpeed(Id, v_{delta})	running	running	$v_{alt} + v_{delta}$	t_{alt}
multiplySpeed(Id, v_{faktor})	running	running	$v_{alt} * v_{faktor}$	t_{alt}
forward(Id)	running	running	$\|v_{alt}\|$	t_{alt}
backward(Id)	running	running	$-\|v_{alt}\|$	t_{alt}
setSpeed(Id, $v_{absolut}$)	running	running	$v_{absolut}$	t_{alt}
activate(Id)	inactive	active	-	-
deactivate(Id)	active	inactive	-	-

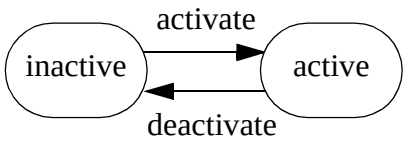
Tabelle 3-1: Elementare Aktionen

In Abbildung 3-6 sind die Elementzustände der Elementarten und die möglichen Zustandsübergänge in Form von Zustandsgraphen zu sehen. Intervalloperatoren und Auswahlgruppen (siehe Abschnitt 3.4.1) haben nur zwei Zustände. Sie sind entweder *active*, dann werden sie in der Präsentation berücksichtigt, oder *inactive*, dann haben sie für die Präsentation keine Relevanz. Für Auswahlgruppen werden die Aktionen *activate* und *deactivate* vom Präsentationssystem bei deren Verarbeitung erzeugt. Reaktionsoperatoren haben noch einen weiteren Zustand *firing* den sie einnehmen, wenn die enthaltenen Aktionen ausgeführt werden. Die Aktionen *triggered* und *fired* werden bei der Abarbeitung eines Reaktionsoperators durch das Präsentationssystem generiert. Interaktionsmanager nehmen den Zustand *sensitive* während ihrer sensitiven Phase ein. Projektoren haben noch weitere Zustände. Im Zustand *running* spielt der Projektor Dateneinheiten aus. Im Zustand *frozen* ist das Ausspielen eingefroren und im Zustand *pausing* wurde

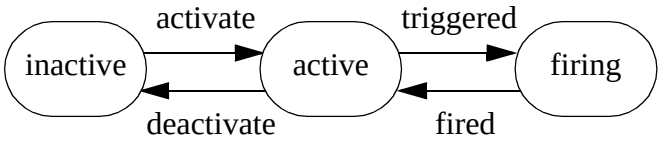
Projektoren:



Intervalloperatoren, Auswahlgruppen:



Reaktionsoperatoren:



Interaktionsmanager:

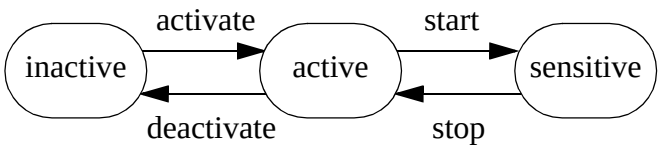


Abb. 3-6: Zustandsgraphen der Dokumentelemente

es unterbrochen. Geschwindigkeitsverändernde Aktionen oder Sprungaktionen sind nur im Zustand *running* erlaubt und führen zu keinem neuen Zustand.

Einige Aktionen können auf mehrere Elementarten angewendet werden, andere nur auf Projektionen. Die Aktionen *activate* und *deactivate* können auf alle Elementarten angewendet werden, außer Auswahlgruppen. Durch sie können Elemente von der weiteren Präsentation ausgeschlos-

sen werden bzw. ausgeschlossene Elemente wieder in die Präsentation eingeschlossen werden. Durch die Deaktivierung eines Intervalloperators kann beispielsweise das zeitliche Layout verändert werden. Wird ein zusammengesetztes Element inaktiv, werden alle in ihm enthaltenen Elemente inaktiv.

Die Aktion *start* bewirkt, daß die Verarbeitungsphase des referenzierten Elements beginnt. Wird ein Projektor angegeben, startet seine Präsentation mit der spezifizierten Präsentationsgeschwindigkeit $v_{spezifiziert}$ am Medienzeitpunkt 0 falls die Ausrichtungspolitik *AlignLength* spezifiziert wurde bzw. am Medienzeitpunkt *Instant* bei den anderen Politiken. Wird ein Interaktionsmanager angegeben, startet seine sensitive Phase. Die Aktion *stop* bewirkt das Beenden der Verarbeitungsphase. Stoppaktionen können im Zustand *frozen*, *running* oder *pausing* auftreten.

Zwischen verschiedenen Interaktionsmöglichkeiten besteht oft ein kausaler Zusammenhang. Beispielsweise kann das Auslösen einer Interaktionsmöglichkeit das Auslösen einer anderen Interaktionsmöglichkeit unmöglich machen. Kausale Abhängigkeiten zwischen Interaktionsmanagern können durch *start*- und *stop*-Aktionen spezifiziert werden. Die Aktionen *start* und *stop* treten nicht nur bei der Abarbeitung von Reaktionsoperatoren auf, sondern werden auch intern generiert, wenn das Präsentationsintervall eines Projektors oder das Interaktionsintervall eines Interaktionsmanagers beginnt bzw. endet.

Die restlichen Aktionen in Tabelle 3-1 beeinflussen die Projektion von Datenräumen. Es gibt Aktionen zur Veränderung der Präsentationsgeschwindigkeit einer Projektion und zur Veränderung der Medienzeit eines Projektors. Wird die Medienzeit eines Projektors manipuliert, wird die Projektion an entsprechender Stelle im Datenraum fortgeführt. Die Aktion *jumpRelative* bewirkt einen Sprung relativ zum aktuellen Medienzeitpunkt um t_{delta} Zeiteinheiten. Je nach Vorzeichen des Parameters wird vorwärts oder rückwärts gesprungen. Demhingegen bewirkt die Aktion *jumpTo* einen Sprung zum Medienzeitpunkt $t_{absolut}$.

Die Aktion *pause* bewirkt, daß die Präsentationsgeschwindigkeit des Projektors auf null gesetzt wird. Dadurch wird die Präsentation der Mediendaten angehalten. Die Aktion *continue* bewirkt, daß die Präsentation der Mediendaten an der gleichen Stelle fortgeführt wird, d.h. die Präsentationsgeschwindigkeit wird auf den vorherigen Geschwindigkeitswert gesetzt. Die Aktion *freeze* bewirkt genau wie *pause*, daß die Präsentation der Mediendaten angehalten wird. Im Gegensatz

zu *continue*, wird durch die Aktion *thaw* die Projektion aber an dem Medienzeitpunkt fortgesetzt an dem man wäre, wenn die Präsentation nie angehalten worden wäre.

Die Aktion *addToSpeed* führt zu einer um den Parameter v_{delta} erhöhten Geschwindigkeit. Ist der Parameterwert negativ, wird die Geschwindigkeit verringert. Bei der Aktion *multiplySpeed* wird die Geschwindigkeit mit dem Parameter v_{faktor} multipliziert. Ist der Parameterwert minus eins wird die Präsentationsgeschwindigkeit negiert, d.h. die Präsentation erfolgt im weiteren rückwärts. Die Aktionen *forward* und *backward* bewirken, daß die Geschwindigkeit positiv bzw. negativ wird, d.h die Präsentationsrichtung kann in eine bestimmte Richtung beeinflußt werden. Die Aktion *setSpeed* setzt die Geschwindigkeit auf den Wert $v_{absolut}$.

Um komplexe Interaktionseffekte zu beschreiben, können **zusammengesetzte Aktionen** spezifiziert werden, die mehrere einfache und andere zusammengesetzte Aktionen gruppieren und festlegen in welcher Reihenfolge diese Aktionen ausgeführt werden. Das Tiempo-Modell bietet zwei zusammengesetzte Aktionen an:

- Die Aktion *SequentialAction*

```
<!ELEMENT SequentialAction (SequentialAction | ParallelAction | Activate | Deactivate |
                            Start | Stop | JumpTo | JumpRelative | Pause | Continue |
                            Freeze | Thaw | AddToSpeed | MultiplySpeed | SetSpeed |
                            Forward | Backward)+ >

<!ATTLIST SequentialAction
      Tiempo      NMTOKEN      #FIXED  "CompositeAction"
>
```

  bewirkt, daß die enthaltenen Aktionen in der spezifizierten Reihenfolge ausgelöst werden. Bei einer *SequentialAction* mit *n* enthaltenen Aktionen tritt dementsprechend eine Abfolge von *n* neuen Kontexten auf:

$$C_i^E \rightarrow C_{i+1}^E \rightarrow \ldots \rightarrow C_{i+n-1}^E \rightarrow C_{i+n}^E \qquad (3\text{-}2)$$

  Diese Kontexte bezeichnet man als externe Kontexte C_j^E, da sie – wenn auch nur kurz – für den Nutzer sichbar sind.

- Die Aktion *ParallelAction*

```
<!ELEMENT ParallelAction (SequentialAction | ParallelAction | Activate | Deactivate |
                          Start | Stop | JumpTo | JumpRelative | Pause | Continue |
                          Freeze | Thaw | AddToSpeed | MultiplySpeed | SetSpeed |
                          Forward | Backward)+ >

<!ATTLIST ParallelAction
      Tiempo      NMTOKEN      #FIXED  "CompositeAction"
>
```

bewirkt, daß die Interaktionseffekte der enthaltenen Aktionen gleichzeitig ausgelöst werden. Bei der Vorbereitung der Effekte werden die enthaltenen Aktionen aber in der spezifizierten Reihenfolge betrachtet. Bei einer *ParallelAction* entsteht, unabhängig von der Anzahl *n* der enthaltenen Aktionen, nur ein neuer externer Kontext:

$$C_i^E \rightarrow C_1^I \rightarrow \ldots \rightarrow C_{n-1}^I \rightarrow C_{i+1}^E \tag{3-3}$$

Bei der sequentiellen Abarbeitung der enthaltenen Aktionen innerhalb des Präsentationssystems entstehen *n*-1 interne Kontexte C_j^I, die der Nutzer nicht sieht.

Mit Hilfe von zusammengesetzten Aktionen verstärkt sich die Ausdrucksfähigkeit hinsichtlich den beschreibbaren Interaktionseffekten erheblich. Um beispielsweise einen Hyperlink zu realisieren, der von einem Dokument zu einem anderen springt, würde man eine sequentielle Aktion wie folgt definieren:

```
<Reaction ...>
      <SequentialAction>
            <Stop  Element="oldDoc" />
            <Start  Element="newDoc" />
      </SequentialAction>
</Reaction>
```

Dadurch würde zuerst das alte Dokument *oldDoc* vom Bildschirm verschwinden und dann das neue Dokument *newDoc* angezeigt werden. Der Bildschirm ist hierbei einen Augenblick leer. Würde man anstelle der sequentiellen eine parallele Aktion verwenden, wäre der Bildschirm nie leer, da das neue Dokument das alte unmittelbar ersetzt.

Um zum Zeitpunkt 23 eines neuen Dokuments zu springen, wobei das bisher präsentierte Dokument beendet wird, ist folgendes zu spezifizieren:

```
<Reaction ...>
    <SequentialAction>
        <Stop  Element="oldDoc" />
        <ParallelAction>
            <Start  Element="newDoc" />
            <JumpTo  Element="newDoc" Instant="23" />
        </ParallelAction>
    </SequentialAction>
</Reaction>
```

Soll nach der Pausierung zweier Medienobjekte *A* und *B* die Präsentation mit der doppelten Geschwindigkeit fortgeführt werden, kann dies wie folgt spezifiziert werden:

```
<Reaction ...>
    <ParallelAction>
        <Continue  Element="A" />
        <MultiplySpeed  Element="A" Factor="2" />
        <Continue  Element="B" />
        <MultiplySpeed  Element="B" Factor="2" />
    </ParallelAction>
</Reaction>
```

3.4 Modellierung von Adaptivität

Das Tiempo-Modell unterstützt die Modellierung von Adaptivität auf zwei Abstraktionsebenen [Wira97, WiRo98]. Adaptivität auf Dokumentelementebene wird durch **Auswahlgruppen** modelliert. Zur Modellierung von Adaptivität auf Attributebene integriert das Modell **Dienstgütebereiche**. Auswahlgruppen und Dienstgütebereiche können kombiniert angewendet werden, um eine optimale Adaptivität von Multimedia-Dokumenten zu spezifizieren.

3.4.1 Auswahlgruppen

Adaptivität auf Dokumentelementebene wird durch folgende Elementtypen modelliert:

```
<!ELEMENT SelectionGroup  (Alternative+) >

<!ATTLIST SelectionGroup
    Tiempo      NMTOKEN                 #FIXED  "SelectionGroup"
    Id          ID                      #REQUIRED
    Policy      (first | static | dynamic)   "static"
>
```

```
<!ELEMENT Alternative  EMPTY >

<!ATTLIST Alternative
        Tiempo      NMTOKEN     #FIXED  "Alternative"
        Priority    CDATA       "0"
        Elements    IDREFS      #IMPLIED
>
```

Eine Auswahlgruppe *SelectionGroup* besteht aus mehreren Präsentationsalternativen *Alternative*. Eine Präsentationsalternative kann aus beliebigen Dokumentelementen bestehen, die durch das Attribut *Elements* referenziert werden, und somit eine komplexe Teilspezifikation repräsentieren. Eine Auswahlgruppe hat die Semantik, daß genau eine der enthaltenen Präsentationsalternativen in der Präsentation realisiert werden muß. Falls Teile einer Präsentation optional sind, kann dies durch Auswahlgruppen mit einer leeren Präsentationsalternative modelliert werden. Präsentationsalternativen können wiederum Auswahlgruppen referenzieren. Auf diese Weise können Auswahlgruppen geschachtelt werden, um Alternativen auf unterschiedlichen Ebenen zu spezifizieren.

Um den Spezifikationsaufwand zu reduzieren, müssen Intervalloperatoren, die in Präsentationsalternativen enthaltene Elemente mit anderen Elementen in Beziehung setzen, nicht in den Präsentationsalternativen enthalten sein, da das Präsentationssystem selbst feststellen kann, ob ein Intervalloperator zu berücksichtigen ist oder nicht. Wird nämlich ein Medium nicht präsentiert, haben alle Intervalloperatoren, die dieses Medium mit anderen Medien in Beziehung setzen, keine Bedeutung und bleiben unberücksichtigt.

Während der Präsentation eines Multimedia-Dokuments kann es passieren, daß mehrere Präsentationsalternativen einer Auswahlgruppe realisiert werden können. Tiempo erlaubt daher, daß Präsentationsalternativen **Prioritäten** zugewiesen werden. Eine höhere Priorität *Priority* impliziert hierbei eine höhere Präsentationsqualität. Optimale Präsentationsalternativen werden aus einer globalen Sicht ausgewählt. Das bedeutet, es werden Präsentationsalternativen aus Auswahlgruppen realisiert, so daß die Summe ihrer Prioritäten maximal ist. Eine lokale Auswahl für jede Auswahlgruppe separat, wie sie beispielsweise in [Hosc98] vorgeschlagen wird, impliziert nämlich die Problematik, daß nicht definiert ist, welcher Auswahlgruppe mehr Ressourcen zugewiesen werden sollen. Um dieses Problem zu beheben, müßte man Auswahlgruppen ebenfalls Prioritäten zuweisen. Diese zweistufige Prioritätsvergabe ist jedoch komplexer

als der Tiempo-Ansatz. Zudem treten hierbei die Präsentationsalternativen und damit die Medien in den Hintergrund und die Auswahlgruppen in den Vordergrund. Zur Bestimmung der Präsentationsqualität sind jedoch die Medien und die durch sie präsentierte Information relevant, und nicht die Auswahlmöglichkeiten an sich.

Mit dem bisher beschriebenen Instrumentarium kann noch nicht definiert werden, wann eine Präsentationsalternative aus einer Auswahlgruppe tatsächlich ausgewählt bzw. eventuell durch eine andere Präsentationsalternative ersetzt werden kann. Zur Kontrolle dieses Aspekts bietet das Tiempo-Modell die folgenden **Auswahlpolitiken** für Auswahlgruppen an, die im Attribut *Policy* spezifiziert werden:

- *first*: Es kann nur einmal während einer gesamten Präsentation eine Präsentationsalternative ausgewählt werden. Wird der Dokumentteil mit der Auswahlgruppe mehrmals präsentiert, wird immer die gleiche, beim ersten Mal gewählte, Präsentationsalternative realisiert.

- *static*: Bei mehrmaligem Präsentieren des Dokumentteils mit der Auswahlgruppe kann jedesmal eine andere Präsentationsalternative realisiert werden. Sobald jedoch mit der Realisierung einer Präsentationsalternative begonnen wurde, darf nicht mehr zu einer anderen Präsentationsalternative gewechselt werden.

- *dynamic*: Es kann jederzeit zwischen den in der Auswahlgruppe enthaltenen Präsentationsalternativen hin und her gewechselt werden.

Abbildung 3-7 zeigt den Teil einer Dokumentspezifikation mit geschachtelten Auswahlgruppen. Die äußere Auswahlgruppe enthält zwei Präsentationsalternativen. Während die erste Präsentationsalternative aus einem Text besteht, enthält die zweite Präsentationsalternative eine Animation und eine weitere Auswahlgruppe. Diese Auswahlgruppe definiert drei Präsentationsalternativen, eine Sprachsequenz, Untertitel und eine leere Präsentationsalternative. Entsprechend der Semantik einer Auswahlgruppe kann hier der Text oder die Animation präsentiert werden. Wenn die Animation präsentiert wird, kann dies mit der Sprachsequenz, den Untertiteln oder ohne zusätzliche Information sein.

Für die äußere Auswahlgruppe ist die Auswahlpolitik *static* spezifiziert. Somit ist es nicht erlaubt, zwischen der Textpräsentation und der Animation zu wechseln, während die Präsenta-

```
<Text  Id="text" ... />
<Animation  Id="anim" ... />
<Audio  Id="rede" ... />
<Subtitel  Id="utitel" ... />

<While  Id="W1" From="anim" To="rede" ... />
<While  Id="W2" From="anim" To="utitel" ... />

<SelectionGroup  Id="A2" Policy="dynamic" >
    <Alternative  Priority="80" Elements="rede" />
    <Alternative  Priority="60" Elements="utitel" />
    <Alternative  Priority="0" />
</SelectionGroup>

<SelectionGroup  Id="A1" Policy="static" >
    <Alternative  Priority="100" Elements="anim A2"/>
    <Alternative  Priority="40" Elements="text" />
</SelectionGroup>
```

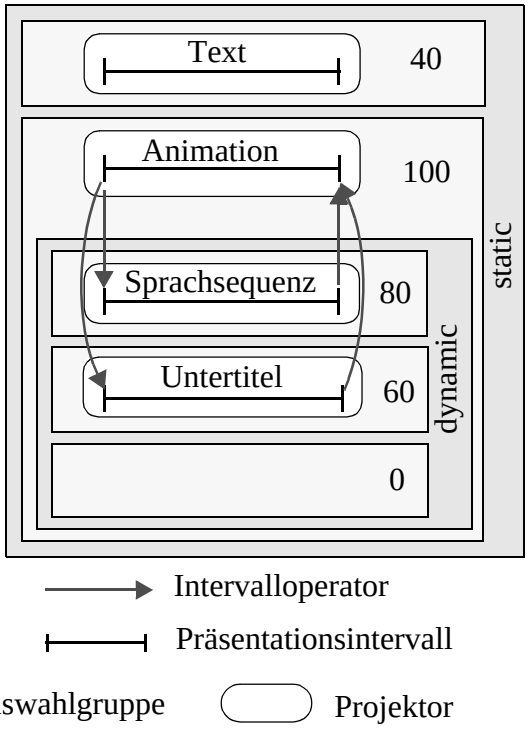


Abb. 3-7: Auswahlgruppenbeispiel

tion der Auswahlgruppe durchgeführt wird. Für die innere Auswahlgruppe ist die Auswahlpolitik *dynamic* spezifiziert, dadurch ist es erlaubt zwischen der Präsentation der Sprachsequenz, den Untertiteln und keiner zusätzlichen Information hin und her zu wechseln, wann immer dies erforderlich ist. So kann beim Auftreten einer Ressourcenknappheit von der Präsentation der Sprachsequenz auf die Untertitel umgeschaltet werden und falls dies noch nicht ausreicht, kann auch das Ausspielen der Untertitel pausiert werden. Wenn im Laufe der Präsentation wieder mehr Ressourcen zur Verfügung stehen, kann dann wieder entweder die Sprachsequenz oder die Untertitel präsentiert werden. In der äußeren Auswahlgruppe hat die Präsentationsalternative mit der Animation die höchste Priorität (100) und wird somit, vorausgesetzt die benötigten Ressourcen sind verfügbar, bevorzugt ausgewählt. In der inneren Auswahlgruppe ist die Sprachsequenz die bevorzugte Präsentationsalternative (Priorität 80), gefolgt von den Untertiteln (Priorität 60) und der leeren Präsentationsalternative (Priorität 0).

Sollen unterschiedliche zeitliche Anordnungen von Medien zugelassen werden, kann dies durch eine Auswahlgruppe spezifizieren werden, die Präsentationsalternativen mit alternativen Inter-

valloperatoren enthält. Will man dem Präsentationssystem die Möglichkeit geben, bestimmte Interaktionen nur anzubieten, wenn genügend Ressourcen da sind um die Interaktionseffekte umzusetzen, kann man eine Auswahlgruppe spezifizieren, die eine Präsentationsalternative mit dem entsprechenden Interaktionsmedium enthält und eine leere Präsentationsalternative. Auswahlgruppen bieten somit vielfältige Möglichkeiten ein Dokument adaptiv zu spezifizieren.

3.4.2 Dienstgütebereiche

Um flexible Attributwerte zu definieren, bietet das Tiempo-Modell Dienstgütebereiche an. Ein Dienstgütebereich repräsentiert ein Werteintervall in dem jedem Wert eine Priorität zugeordnet ist. Bei einer Dokumentpräsentation kann das Präsentationssystem einen Wert aus dem Dienstgütebereich wählen. Die Priorität eines Werts beschreibt wie wichtig der Wert für die Präsentation ist.

Ein Dienstgütebereich hat die Form $[w_1{:}p_1, w_2{:}p_2, w_3{:}p_3, \ldots, w_k{:}p_k]{:}g$. Er definiert das Werteintervall $[w_1, w_k]$. Die Wertepaare repräsentieren Stützstellen, welche eine Prioritätsstruktur beschreiben. Ein Wertepaar $w_i{:}p_i$ weist dem Wert w_i die Priorität p_i zu. Für die Werte w zwischen zwei Stützstellen $w_i{:}p_i$ und $w_{i+1}{:}p_{i+1}$ wird die Priorität $P(w)$ durch lineare Funktionen der Form

$$P(w) = \frac{p_{i+1} - p_i}{w_{i+1} - w_i}(w - w_i) + p_i, \qquad w_i \le w \le w_{i+1} \tag{3-4}$$

definiert. Da Dienstgütebereiche beliebige Stützstellen enthalten können, kann der Autor beliebig detaillierte Prioritätsstrukturen spezifizieren. Um zu definieren wann ein Wert aus einem Dienstgütebereich ausgewählt werden darf, können einem Dienstgütebereich durch den Parameter *g* die gleichen Auswahlpolitiken wie Auswahlgruppen zugewiesen werden. Der Parameter *g* kann die Werte *f* für *first*, *d* für *dynamic* und *s* für *static* annehmen. Sollen keine Prioritäten oder Auswahlpolitiken spezifiziert werden, läßt man die Angaben weg. Die Standardpriorität ist null und die Standardauswahlpolitik ist *dynamic*.

Abbildung 3-8 zeigt exemplarisch die durch einen Dienstgütebereich definierte Prioritätsstruktur. Im Beispiel hat der Dienstgütebereich vier Stützstellen. Diese definieren einen Wertebereich zwischen 10 und 55 Sekunden. Hierbei hat der Wert 10 die Priorität 30, der Wert 22 die

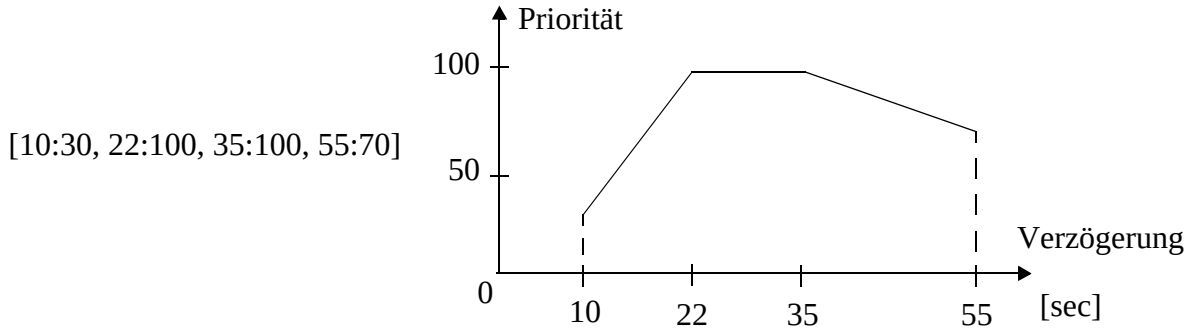


Abb. 3-8: Dienstgütebereichbeispiel

Priorität 100, der Wert 35 die Priorität 100 und der Wert 55 die Priorität 70. Entsprechend der Semantik eines Dienstgütebereichs werden mit den Werten zwischen 10 und 22 linear ansteigende Prioritäten von 30 bis 100 assoziiert. Mit den Werten zwischen 35 und 55 werden linear absteigende Prioritäten von 100 bis 70 assoziiert und die Werte zwischen 22 und 35 haben alle die Priorität 100. Hier sollte das Präsentationssystem, um die optimale Präsentationsqualität zu erreichen, einen Wert zwischen 22 und 35 wählen.

Dienstgütebereiche können in Tiempo in Projektionselementen zur Spezifikation der Präsentationsintervallänge und Präsentationsrate verwendet werden und in Intervalloperatoren zur Spezifikation der Verzögerungsparameter. Des weiteren können Dienstgütebereiche in Interaktionsmanagern zur Spezifikation der Interaktionsintervallänge sowie der Attribute *StartDelay* und *EndDelay* verwendet werden.

Die Unterscheidung zwischen statischer und dynamischer Auswahlpolitik ist nur bei Dienstgütebereichen, die eine Präsentationsrate definieren, sinnvoll, da eine entsprechende dynamische Veränderung vom Nutzer wahrgenommen wird. Werden Dienstgütebereiche zur Spezifikation von Intervallängen oder Verzögerungen eingesetzt, haben beide Auswahlpolitiken aus Sicht des Nutzers denselben Effekt. Die Anwendung der Auswahlpolitik *first* macht jedoch Sinn, unabhängig davon für welches Attribut ein Dienstgütebereich eingesetzt wird.

Die folgenden Beispiele sollen die vielfältigen Möglichkeiten aufzeigen, die Projektionen in Kombination mit Dienstgütebereichen schaffen, um adaptive Dokumente zu spezifizieren. Im folgenden Dokumentausschnitt wird ein Fenster definiert in dem zwei Videos überlappend ausgespielt werden. Die Videos enden simultan. Video *V2* wird komplett präsentiert, da die Aus-

richtungspolitik *AlignLength* spezifiziert wurde, dabei kann seine Präsentationsrate zwischen 12 und 15 Bildern/Sekunde variieren. Video *V1* startet zwischen 5 und 20 Sekunden nach Video *V2*. Die Ausrichtungspolitik *ForceEnd* für Video *V1* ist so spezifiziert, daß unabhängig davon wann seine Präsentation beginnt, es immer mit dem Medienzeitpunkt 110 beendet wird.

```
<Window  Id="F" ...>
        <VideoPro  Id="V1" >
                <Projection  PresInterval="[0,?]" Speed="1.0" Rate="1.0" >
                        <ForceEnd  Instant="110" Position="last" />
                </Projection>
                <Video  Length="120" Rate="20" .../>
        </VideoPro>
        <VideoPro  Id="V2" ...>
                <Projection  PresInterval="[0,?]" Speed="1" Rate="[0.8:0,1.0:30]:s" >
                        <AlignLength/>
                </Projection>
                <Video  Length="100" Rate="15" .../>
        </VideoPro>
        <While  Id="W" From="V2" To="V1" Delay1="[5:25,20:0]" Delay2="0" />
</Window>
```

Hier würde bei einem Ressourcenengpaß zuerst der Start von Video *V1* hinausgezögert werden und falls dies nicht reicht oder nicht mehr möglich ist, die Präsentationsrate von Video *V2* gesenkt werden.

Sollen zwei Videos nacheinander präsentiert werden, wobei das erste Video *V1* wenig und das zweite Video *V2* viel Ressourcen benötigt und soll von *V1* eigentlich nur ein 30 Sekunden langer Ausschnitt präsentiert werden, dessen Mitte beim Medienzeitpunkt 60 liegt, könnte man wie folgt eine adaptive Präsentation spezifizieren:

```
<Window ...>
        <VideoPro  Id="V1" >
                <Projection  PresInterval="[30:30,70:0]" Speed="1.0" Rate="1.0" >
                        <ForceEnd  Instant="60" Position="center" />
                </Projection>
                <Video  Length="120" Rate="10" Width="150" Height="150" .../>
        </VideoPro>
        <VideoPro  Id="V2" ...>
                <Projection  PresInterval="[350:0,400:40]" Speed="1.0" Rate="[0.7:0,1.0:20]:s" >
                        <ForceEnd  Instant="0" Position="first" />
                </Projection>
                <Video  Length="400" Rate="25" Width="400" Height="300" .../>
```

```
        </VideoPro>
        <Before  Id="W" From="V1" To="V2" Delay1="0" />
</Window>
```

Die entsprechend spezifizierte Ausrichtungspolitik *ForceEnd* von *V1* sorgt dafür, daß der gewünschte Ausschnitt von *V1* immer in der Mitte der Video-Präsentation gezeigt wird, welche zwischen 30 und 70 Sekunden variieren kann. Dadurch kann nun die Präsentation des zweiten Videos *V2* bis zu 40 Sekunden verzögert werden, wenn momentan eine Ressourcenknappheit herrscht. Während des Ausspielens von *V2* kann falls erforderlich seine Rate gesenkt werden, und wenn das nicht ausreicht, kann seine Präsentation 50 Sekunden vor dem Ende abgebrochen werden.

Wie diese Beispiele zeigen, hat ein Autor durch die unterschiedlichen Ausrichtungspolitiken in Kombination mit Dienstgütebereichen einerseits optimale Möglichkeiten Flexibilität in ein Dokument zu integrieren und andererseits kann er kontrollieren welche Medienteile tatsächlich präsentiert werden.

3.5 Zusammenfassung und Bewertung

Das Tiempo-Dokumentenmodell erlaubt die Modellierung adaptiver Multimedia-Dokumente. Das Konzept der geschachtelten Datenräume in Kombination mit den Intervalloperatoren erlaubt es, alle relevanten zeitlichen Abhängigkeiten zwischen Medien zu beschreiben, und verglichen mit Firefly [BuZe92, BuZe93] und CHIMP [CPS96], übersichtliche hierarchisch-strukturierte Multimedia-Dokumente zu erzeugen. Durch Projektionen wird es ermöglicht, Medien bzw. Medienteile neu zusammenzustellen und mit unterschiedlicher Charakteristik zu präsentieren. Das Interaktionsmodell von Tiempo erlaubt die Spezifikation von komplexen Benutzerinteraktionen.

Das Tiempo-Modell unterstützt die in Abschnitt 2.2 identifizierten Möglichkeiten zur Flexibilisierung von Multimedia-Dokumenten, mit Ausnahme der expliziten Spezifikation von Jitter und Skew. Hierauf wurde mit Absicht verzichtet, da deren Spezifikation beim Autor eines Dokuments ein detailliertes Wissen über Stromsynchronisationsaspekte erfordern würde. Die meisten Autoren von Multimedia-Präsentation sind jedoch Designer oder Werbefachleute,

denen derartige Techniken nicht geläufig sind. Das Tiempo-Modell macht implizite Annahmen über den einzuhaltenden Jitter und Skew entsprechend der Untersuchung in [StEn93].

Im Vergleich zu Mbuild [HSR92, HSR94], Firefly und CHIMP erlaubt es das Tiempo-Modell, für alle Attribute, die mit zeitlichen Aspekten korreliert sind, Präsentationsalternativen mit Hilfe von Dienstgütebereichen zu spezifizieren, insbesondere auch für Präsentationsraten. Im Vergleich zu MODE [BHL91] erlaubt es Tiempo nicht nur einen bevorzugten Wert aus einem Wertebereich zu identifizieren, sondern jedem Wert eine Priorität zuzuweisen. Damit ist die maximal mögliche Granularität bei der Priorisierung von Präsentationsalternativen gegeben.

Hinsichtlich der Spezifikation von Adaptivität auf Objektebene ist der Tiempo-Ansatz den Modellen Mbuild und Firefly überlegen, da Mbuild und Firefly diese Art von Adaptivität nicht unterstützen. Tiempo ist auch CHIMP und MODE überlegen, da diese Ansätze nicht alle Arten von Adaptivität auf Objektebene unterstützen.

Das Tiempo-Modell erlaubt auf Attributebene und Objektebene die Spezifikation von Auswahlpolitiken, wodurch festgelegt werden kann, wann eine Adaption gewünscht ist. Dies ist bei keinem der existierenden Ansätze möglich.

Durch die Anwendung von Projektionen in Kombination mit Dienstgütebereichen kann im Detail festgelegt werden, welche Medienteile bei Adaptionen weggelassen werden. Dadurch wird es möglich mehr Flexibilität in ein Dokument zu integrieren. Bei existierenden Ansätzen kann dies nicht spezifiziert werden, es wird immer das Ende der Präsentation weggelassen.

Das Tiempo-Modell integriert damit einen umfassenden Ansatz für die Modellierung adaptiver Multimedia-Dokumente.

4 Spezifikation adaptiver Multimedia-Dokumente

Multimedia-Präsentationen werden häufig von Werbefachleuten, Designern oder Pädagogen entwickelt, die kaum Programmierkenntnisse besitzen. Diesen Personen müssen Software-Werkzeuge zur Verfügung gestellt werden, welche die Erstellung von multimedialen Präsentationen durch visuelle Programmiertechniken ermöglichen und weitestgehend ohne textuelle Programmiersprachen auskommen. Solche graphisch-interaktive Werkzeuge werden **Autorensysteme** genannt [EHV93]. Um die Aufgaben eines Autorensystems darzustellen, betrachten wir in Abschnitt 4.1 den Entwicklungsprozeß einer multimedialen Präsentation. In Abschnitt 4.2 werden Ansätze vorgestellt, auf denen existierende Autorensysteme basieren und anschließend neue Anforderungen im Bezug auf adaptive Präsentationen erörtert. Danach wird die Konzeption des Tiempo-Autorensystems vorgestellt.

4.1 Entwicklungsprozeß multimedialer Präsentationen

Die Entwicklung einer interaktiven multimedialen Präsentation kann in mehrere Teilschritte zerlegt werden [Bole95]. Der Zusammenhang der einzelnen Schritte ist auf der rechten Seite von Abbildung 4-1 dargestellt. In der **Mediengenerierungsphase** werden elementare Medien mit Hilfe von Mediengenerierungswerkzeugen, wie beispielsweise Graphikeditoren, Texteditoren oder Editoren zur Videobearbeitung, generiert. Auf diese Weise entsteht eine Mediensammlung noch unabhängiger Medien. In der sich anschließenden **Kompositionsphase** werden die unabhängigen Medien zu einer multimedialen Präsentation zusammengefügt. Die Kompositionsphase besteht aus drei Teilphasen: In einer **Layout-Definitionsphase** werden die initialen Eigenschaften der einzelnen Medien bestimmt, wie beispielsweise ihre Position auf dem Bildschirm oder ihre räumliche Überlappung. In einer **Strukturierungsphase** werden die Beziehungen zwischen Medien definiert. In der **Testphase** kann sich ein Autor Ausschnitte der Präsentation anschauen und somit überprüfen, ob die Präsentation seinen Vorstellungen entspricht. Diese Phasen können wiederholt durchlaufen werden, bis der Autor mit dem erstellten Multimedia-Dokument zufrieden ist.

Soll ein adaptives Dokument erstellt werden, müssen in der Layout- und Strukturierungsphase Präsentationsalternativen definiert und in der Testphase überprüft werden.

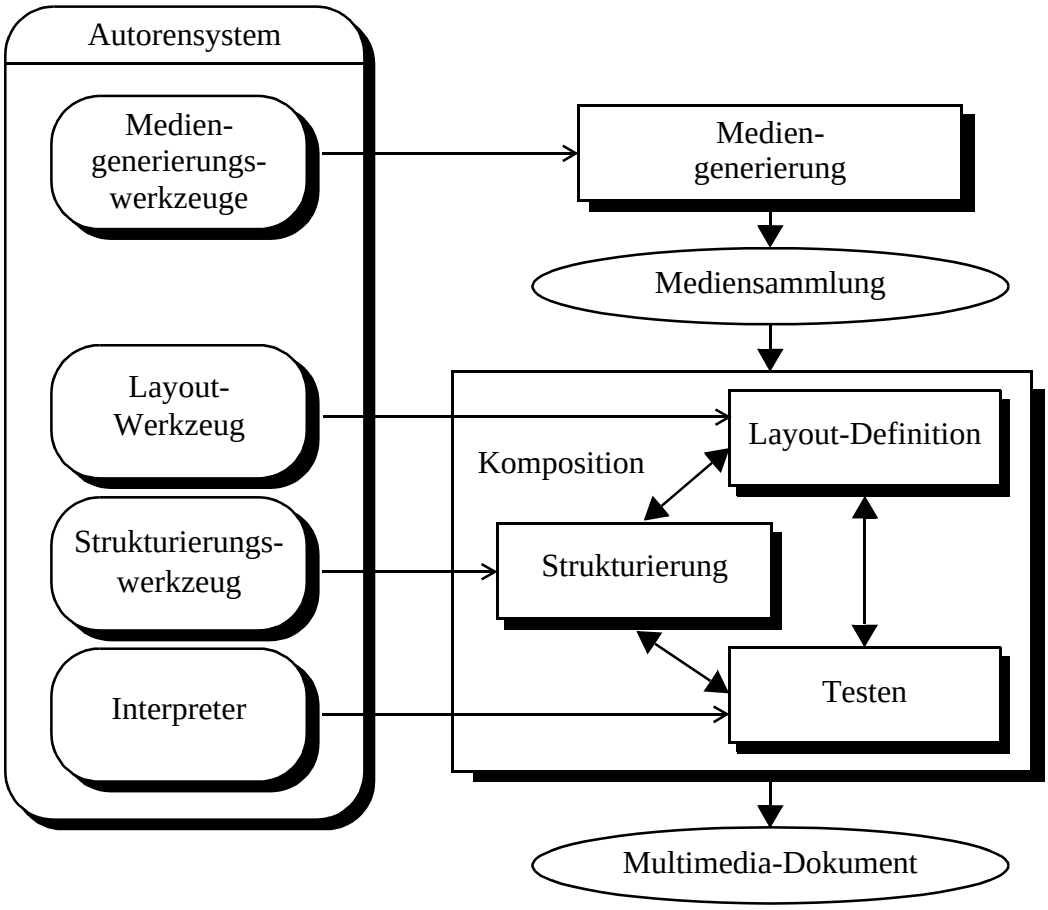


Abb. 4-1 : Entwicklung interaktiver Multimedia-Präsentationen

4.2 Ansätze zur graphisch-interaktiven Spezifikation von Dokumenten

Autorensysteme unterstützen einen Autor bei der Entwicklung multimedialer Präsentationen. Unter Berücksichtigung des in Abschnitt 4.1 beschriebenen Entwicklungsprozesses für multimediale Präsentationen sollte ein Autorensystem für jede Entwicklungsphase entsprechende Werkzeuge zur Verfügung stellen (siehe Abbildung 4-1), d.h. graphisch-interaktive Layout-Werkzeuge, graphisch-interaktive Strukturierungswerkzeuge und einen Interpreter [Bole95]. Eigene Mediengenerierungswerkzeuge müssen nicht bereitgestellt werden, falls externe Mediengenerierungswerkzeuge über Schnittstellen mit dem Autorensystem gekoppelt werden können oder mittels diesen Werkzeugen generierte Medien eingelesen werden können. Welche Technik auch immer zum Einsatz kommt, bei adaptiven Dokumenten müssen abhängig vom verwendeten Ablaufplanungsverfahren gewisse Informationen über den Ressourcenbedarf von Medien während der Präsentation vorhanden sein (siehe Abschnitt 6).

In den letzten Jahren sind eine Vielzahl von Autorensystemen erschienen. Alle Autorensysteme verfolgen den Ansatz der Visualisierung der Präsentationsstruktur. Die meisten Autorensysteme unterstützen dabei lediglich die Modellierung von zeitlichen und räumlichen Beziehungen und von (einfachen) Navigationsinteraktionen. Komplexes Interaktionsverhalten muß meist mittels einer integrierten Skript- oder Programmiersprache spezifiziert werden. Bezüglich der Art und Weise wie Beziehungen zwischen Medien spezifizierbar sind, können existierende Autorensysteme in drei Klassen eingeteilt werden: Bildschirm-basierte, Zeitachsen-basierte und Flowchart-basierte Autorensysteme.

Bei **Bildschirm-basierten Autorensystemen**, wie HyperCard [Good90], ToolBook [Boga92] oder auch ShareME [Vään92], werden die zu präsentierenden Medien auf Flächen (Karten, Seiten oder auch Dias genannt) plaziert. Diese Flächen repräsentieren einen Bildschirm, wie ihn der Benutzer während der Präsentation für einen bestimmten Zeitabschnitt zu sehen bekommt. Eine komplette multimediale Präsentation setzt sich aus einer Menge solcher Flächen zusammen, die einem Benutzer in einer bestimmten von ihm durch Interaktion beeinflußbaren Reihenfolge gezeigt werden. Die Menge der Medien einer solchen Fläche bildet dabei ein zusammengesetztes Medium. Navigationsinteraktionen bewirken im allgemeinen einen Wechsel der gerade präsentierten Karte oder Seite. Insbesondere bei den Bildschirm-basierten Autorensystemen sind die Grenzen zwischen Hypertext- und Hypermedia-Systemen fließend. Temporale Beziehungen zwischen Medien können meist nicht spezifiziert werden.

Bei **Zeitachsen-basierten Autorensystemen**, wie MacroMind Director [Lütj94], MODE [BHL91], Muse [HSA89], MAEstro [Drap93] oder Mbuild [HSR92, HSR94], werden die Medien auf einer Zeitachse angeordnet, die den zeitlichen Verlauf der Präsentation festlegt. Problematisch bei diesen Autorensystemen ist, daß Interaktionen meist nur für die komplette Präsentation definiert werden können, aber nicht für einzelne Medien. Zudem ist für die Spezifikation des räumlichen Layouts ein zusätzliches Konzept erforderlich.

Bei **Flowchart-basierten Autorensystemen**, wie Authorware Professional [Stei93a], dem Apple Media Kit [Volg94], Eventor [ENK+94], XMAD [EiHo93] oder Firefly [BuZe92, BuZe93], werden Medien in Diagrammen durch Icons repräsentiert. Kanten, welche die Icons miteinander verbinden, können eine unterschiedliche Semantik haben. Sie können beispielsweise den möglichen Verlauf der Präsentation widerspiegeln, indem sie Hyperlinks repräsen-

tieren. Gehen dabei von einem Objekt mehrere Kanten zu unterschiedlichen Objekten aus, so wird die während der Präsentation tatsächlich traversierte Kante im allgemeinen durch eine Navigationsinteraktion bestimmt. Sie können jedoch auch räumliche oder zeitliche Beziehungen zwischen Medien repräsentieren.

Einige Autorensysteme, wie beispielsweise HyperCard, ToolBook oder MacroMind Director, stellen eine spezielle Skriptsprache bereit, mit der nicht visualisierte Aspekte der Dokument-Spezifikation programmiert werden können [West91]. Das bedeutet jedoch, daß ein Autor Programmierkenntnisse besitzen muß, was der eigentlichen Motivation von Autorensystemen, daß mit ihrer Hilfe auch Nicht-Programmierer Präsentationen entwickeln können, widerspricht.

4.3 Anforderungen durch die Adaptivität von Dokumenten

Autorensysteme für adaptive Multimedia-Dokumente sollten eine graphisch-interaktive Spezifikation von Präsentationsalternativen erlauben. Um die Übersichtlichkeit zu gewährleisten, sollte ein Autorensystem die individuelle Auswahl, Darstellung und Manipulation jeder validen Kombination von Präsentationsalternativen ermöglichen. Dies gilt insbesondere beim Einsatz wenig abstrakter Visualisierungsmethoden, wie dem Zeitachsen-basierten oder Bildschirm-basierten Ansatz. Betrachtet man beispielsweise den Bildschirm-basierten Ansatz, der aufgrund der WYSIWYG[1]-Darstellung sehr gut zur Spezifikation des räumlichen Layouts geeignet ist, in einem Autorensystem ohne Auswahlmöglichkeiten. Dann führen mehrere Präsentationsalternativen, die an der gleichen Stelle plaziert werden sollen, zu einer sehr unübersichtlichen Darstellung. Eine Simulationskomponente mit der die Auswahl von Präsentationsalternativen während der Präsentation durchgespielt werden kann, unterstützt zusätzlich die Spezifikation von adaptiven Aspekten.

Im Hinblick auf die Testphase bieten existierende Autorensysteme meist keine spezielle Unterstützung an. Eine Spezifikation wird getestet, indem sie mit dem entsprechenden Präsentationssystem präsentiert wird. Im Falle von adaptiven Multimedia-Dokumenten wirft diese Vorgehensweise ein Problem auf: Wie soll man alle möglichen Präsentationsvarianten testen, die ein adaptives Dokument ermöglicht? Um dies zu bewerkstelligen, müßte man die entsprechenden

[1] What You See Is What You Get

unterschiedlichen Ressourcensituationen simulieren, was so gut wie unmöglich ist. Abhilfe für dieses Problem schafft eine automatische Konsistenzprüfung, die sicherstellt, daß erstellte Spezifikationen keine Fehler enthalten und alle Präsentationsvarianten präsentierbar sind.

4.4 Architektur der Benutzerschnittstelle des Tiempo-Editors

Bevor die Mechanismen zur graphisch-interaktiven Spezifikation von Multimedia-Dokumenten beschrieben werden, wird die Architektur der Benutzerschnittstelle des Tiempo-Autorensystems bzw. Tiempo-Editors [Wir99a] vorgestellt.

Nach dem Starten des Tiempo-Editors erscheint das **Editor-Hauptfenster** auf dem Bildschirm. In Abbildung 4-2 ist das Hauptfenster in der linken oberen Ecke zu sehen. Im Hauptfenster befindet sich eine **Zwischenablage**, in der die gerade in Bearbeitung befindlichen Dokumente in Form von Icons angezeigt werden. Die Zwischenablage kann jedoch nicht nur Dokumente enthalten, sondern beliebige Dokumentelemente, wie Videoelemente oder Teildokumente. Die Zwischenablage dient somit als universeller Zwischenspeicher, in dem alle Elemente enthalten sind, die in keinem Dokument enthalten sind. Wird beispielsweise eine Cut&Paste Operation durchgeführt, befinden sich die Dokumentelemente, auf die eine Cut-Operation angewendet wurde, bis zur Paste-Operation in der Zwischenablage.

Alle Elementtypen der Tiempo-Sprache, aus deren Instanzen ein Tiempo-Dokument besteht, werden in sogenannten **Type Selection Panels** (TSP) in Form von Icons visualisiert. Elementtypen mit ähnlicher Semantik werden hierbei im gleichen TSP angeordnet. Es gibt jeweils ein TSP für Medienelemente, Projektoren, Operatoren, Aktionen und Auswahlgruppen. In Abbildung 4-2 sind auf der rechten Seite die TSP für Projektoren, Operatoren und Aktionen zu sehen.

Um dem Autor die Nutzung durch Mediengenerierungswerkzeuge erzeugter Medienobjekte zu vereinfachen, können Verzeichnisse eines Dateisystems durch sogenannte **File Selection Panels** (FSP) visualisiert werden. Auf das Dateisystem kann hierbei entweder über ein Network-File-System (NFS) oder HTTP [FGM+97] zugegriffen werden, wodurch der Editor nicht auf dem gleichen Computer laufen muß, auf dem sich die Medienelemente befinden. Ein FSP stellt die in einem Verzeichnis enthaltenen Dateien je nach Typ durch unterschiedliche Icons dar. Die charakteristischen Attribute eines Medientyps inklusive der Ressourcenanforderungen

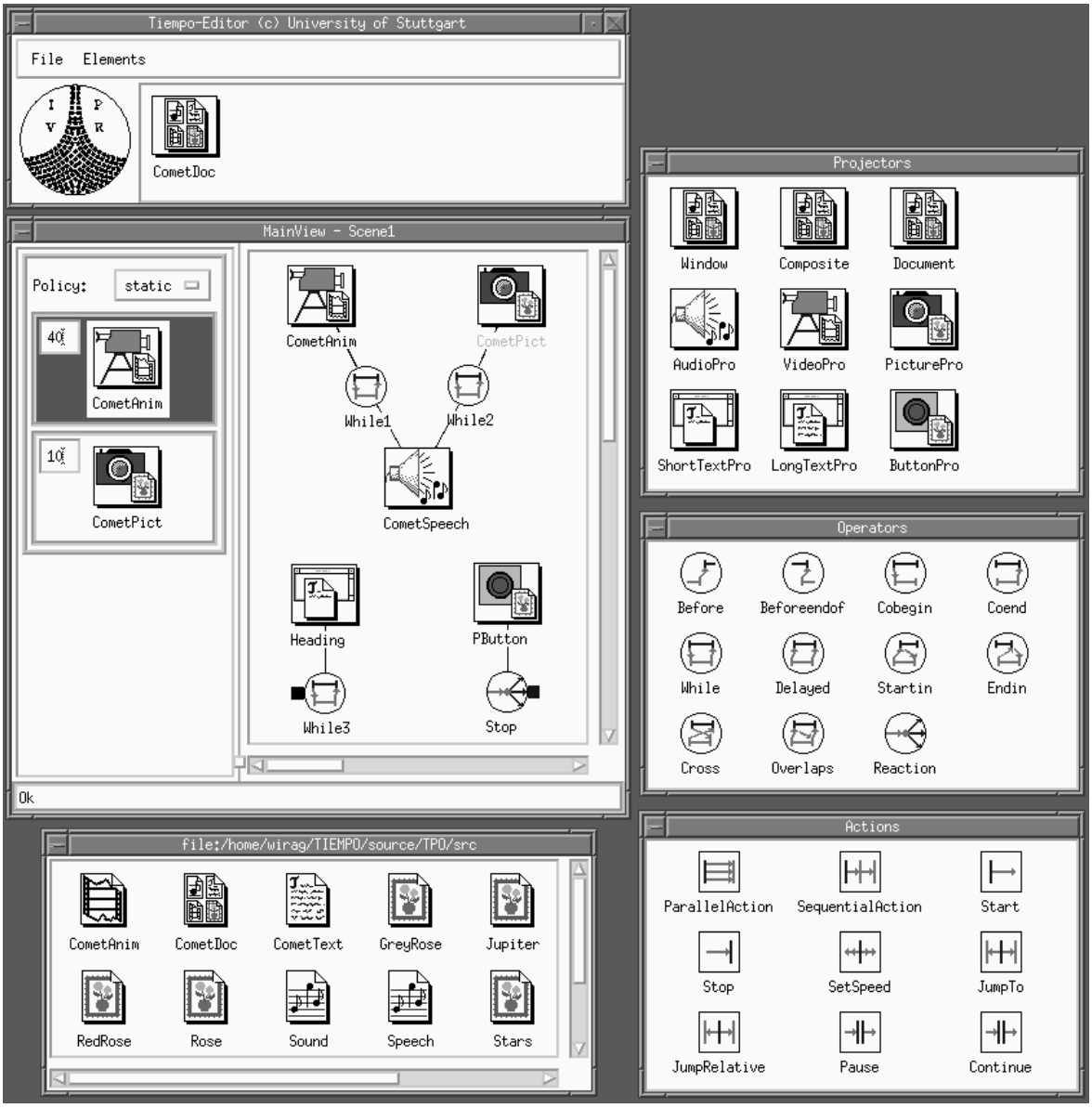


Abb. 4-2 : Graphisch-interaktive Komposition von Dokumenten

werden selbständig vom Editor erkannt. Die aktuelle Version des Editors unterstützt eine solche Erkennung für Dateien mit Bildern in JPEG-, XPM- und GIF-Kodierung, für Videos in MPEG- und Motion-JPEG-Kodierung, für Audio-Dateien im MP3- und WAV-Format und für HTML-Texte. In Abbildung 4-2 ist unten links ein FSP mit Medienelementen zu sehen.

Der Tiempo-Editor ist objektorientiert gestaltet. Graphische Elemente, die Dokumentelemente repräsentieren, besitzen ein Kontextmenü über das Operationen auf dem Element ausgeführt werden können. In TSPs und FSPs enthaltene Objekte besitzen ein Kontextmenü mittels dem sich der Autor deren Eigenschaften anzeigen lassen kann. In der oberen Hälfte von

Abbildung 4-3 ist ein Fenster zu sehen, das die Attribute des Intervalloperators *before* aus dem TSP mit Operatoren anzeigt. In der unteren Hälfte ist ein Fenster zu sehen, daß die Attribute des Medienelements *CometAnim.mpg* aus einem FSP anzeigt. Im Eigenschaftsfenster des *before*-Operators sind die entsprechenden Standardattributwerte des Elementtyps zu sehen. Im Gegensatz dazu sind im Eigenschaftsfenster des Medienelements die Attributwerte zu sehen, die der Editor aus der entsprechenden Mediendatei extrahiert hat, wie beispielsweise die Länge des Video-Clips und seine natürliche Präsentationsrate.

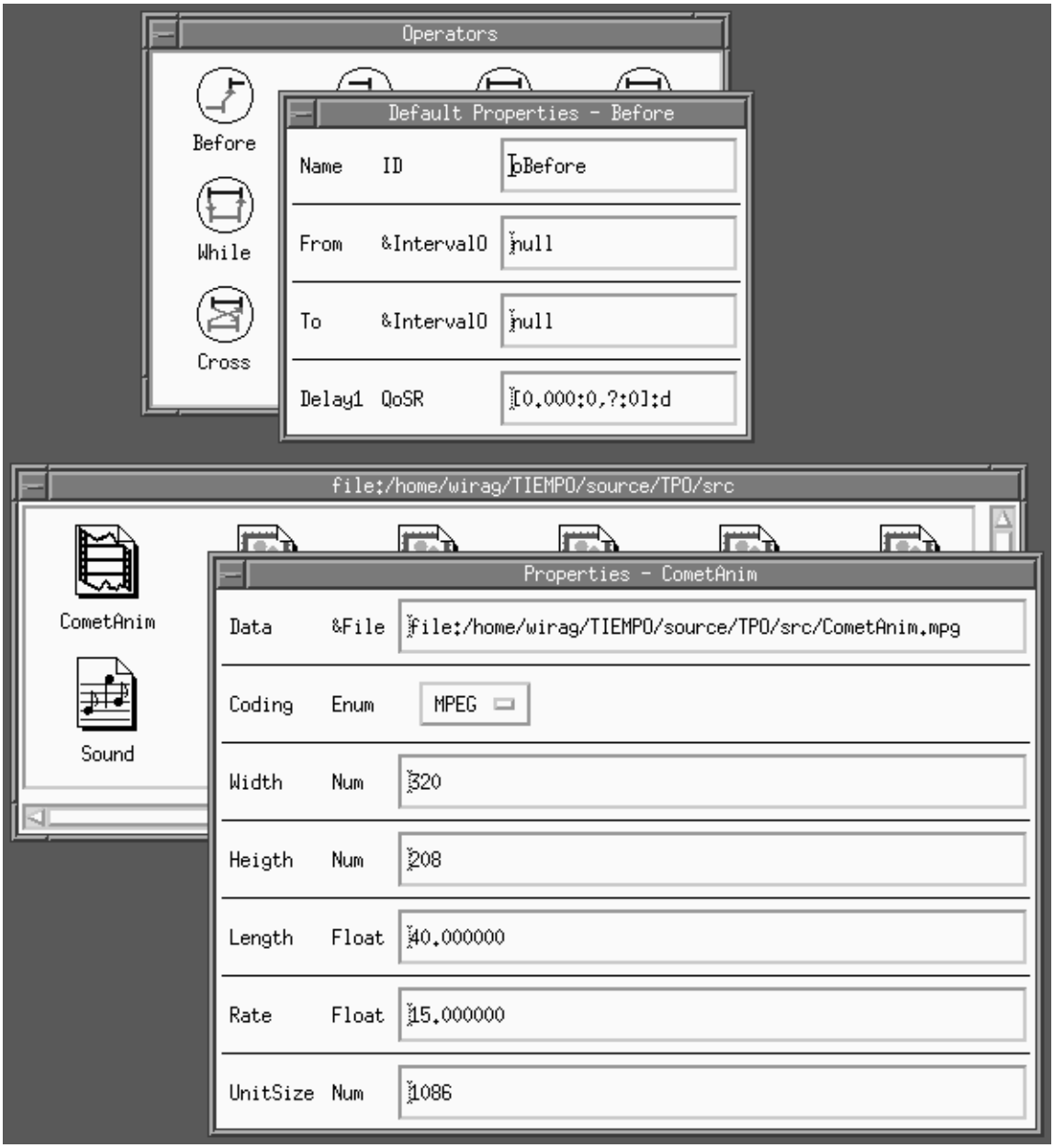


Abb. 4-3 : Visualisierung von Elementeigenschaften

Bei der Spezifikation von Dokumenten wird eine Elementinstanz generiert, indem mittels Drag&Drop das entsprechende Icon aus einem Type Selection Panel oder einem File Selection Panel in das zu einem Dokument gehörende Fenster gezogen wird (siehe Abbildung 4-2 Mitte linke Seite). Bei der Drop-Operation wird dann eine entsprechende Elementinstanz erzeugt. Werden Intervalloperatoren, Reaktionsoperatoren oder Präsentationsalternativen erzeugt, muß nach der Generierung der Elemente durch Drag&Drop noch die Verknüpfung zu anderen Elementen hergestellt werden. Auf diese Weise kann recht schnell die Skizze einer Präsentation erstellt werden. Details können dann beispielsweise über das Setzen von Attributen im Kontextmenü von Elementen spezifiziert werden.

Der Editor unterstützt Cut, Copy und Paste-Operationen, bei der die entsprechenden Elemente in der Zwischenablage zwischengespeichert werden und an anderer Stelle mittels Drag&Drop wieder eingefügt werden können.

Der vielfältige Einsatz von Drag&Drop und die strenge Objektorientierung hat den Vorteil, daß sich der Benutzer des Tiempo-Editors nur wenige Interaktionsschemata merken muß, die prinzipiell immer in gleicher Weise funktionieren. Dadurch kann ein Benutzer in kurzer Zeit den Einsatz des Editors erlernen.

4.5 Strukturierung eines Multimedia-Dokuments

Jedes Element der Tiempo-Sprache, das andere Elemente enthalten kann, besitzt im Tiempo-Editor eine sogenannte **Hauptansicht** (MainView). In eine Hauptansicht können, in Abhängigkeit des Elements, das sie repräsentiert, durch Drag&Drop andere Dokumentelemente eingefügt werden. Die eingefügten Elemente werden durch Icons repräsentiert, die je nach Elementtyp unterschiedlich sind. Auf diese Weise wird die hierarchische Struktur des Multimedia-Dokuments respektive die Schachtelung der Dokumentelemente spezifiziert. Eine Hauptansicht repräsentiert einen Flowchart-basierten Spezifikationsansatz. In Abbildung 4-4 sind die geöffneten Hauptansichten eines Dokuments zu sehen.

Die Hauptansichten von Multimedia-Projektorelementen sind zweigeteilt. Auf der rechten Seite der Hauptansicht sind die enthaltenen Dokumentelemente und deren Verknüpfung durch Operatoren zu sehen. Auf der linken Seite werden die enthaltenen Auswahlgruppen dargestellt.

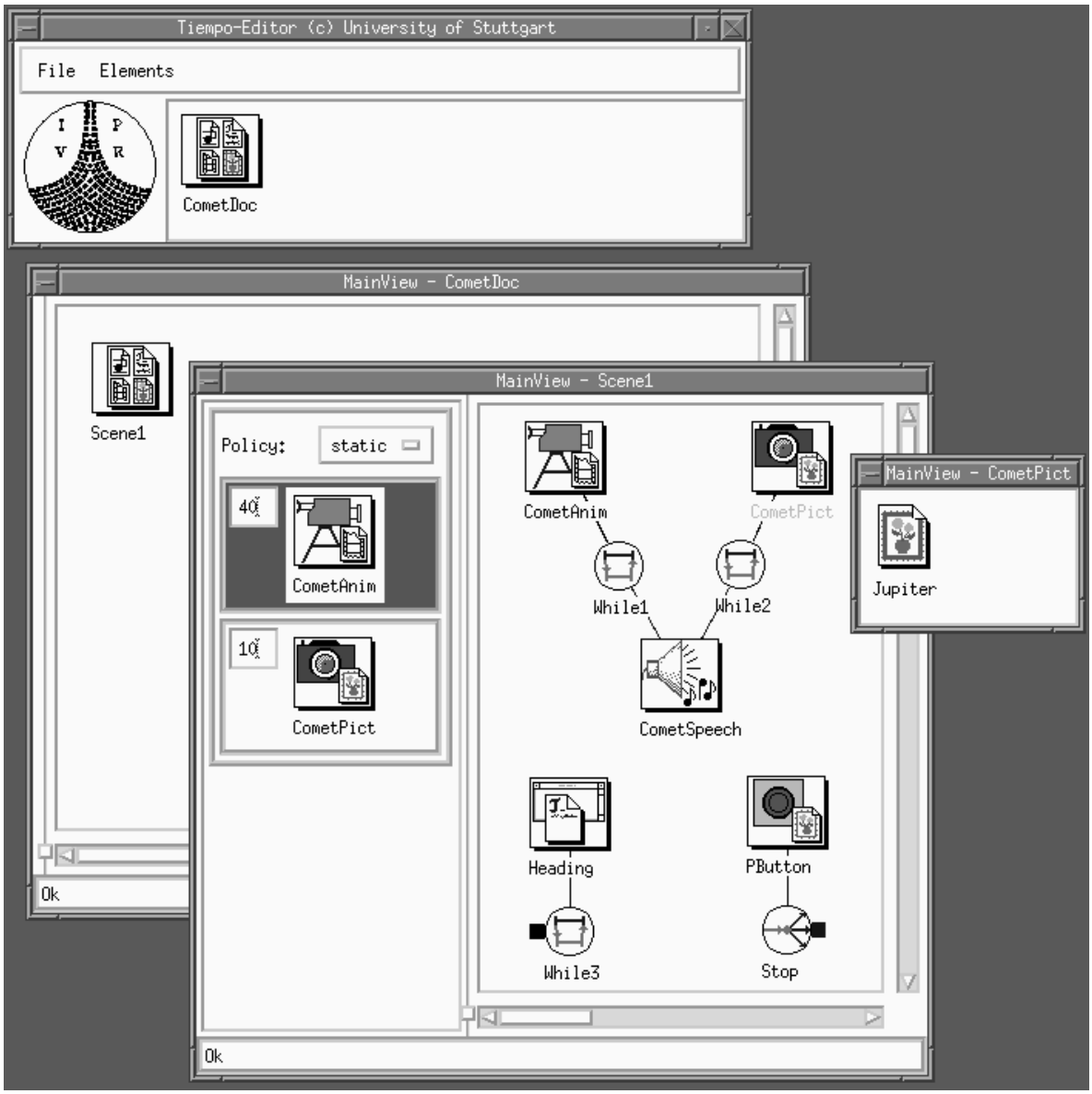


Abb. 4-4 : Hierarchische Strukturierung von Dokumenten

Zur Spezifikation von Beziehungen zwischen Dokumentelementen werden Operatorelemente (Intervalloperatoren und Reaktionsoperatoren) per Drag&Drop in die Hauptansicht eines Multimedia-Projektorelements eingefügt. Ein Operator wird durch ein rundes Icon repräsentiert. Ein Operatorelement besitzt Attribute, mit denen es andere Dokumentelemente referenziert. Diese Referenzattribute werden in der Hauptansicht visualisiert, so daß der Autor die Abhängigkeiten zwischen Elementen sofort erkennen kann. Ist ein Referenzattribut nicht mit einem Wert belegt, besitzt das Icon einen leeren Kreis, der das Attribut repräsentiert. Hat der Autor das Attribut mit einem entsprechenden Dokumentelement verknüpft, das im selben Multimedia-

Projektorelement wie der Operator enthalten ist, wird anstelle des Kreises eine Linie zwischen Operator und Element gezogen. Verknüpft der Autor das Attribut mit einem Dokumentelement in einem anderen Multimedia-Projektorelement, wird das Attribut durch einen gefüllten Kreis dargestellt. Verknüpft der Autor ein Referenzattribut mit dem umgebenden Multimedia-Projektorelement, wird das Attribut durch ein Quadrat repräsentiert. Damit der Autor unterschiedliche Referenzattribute eines Operators unterscheiden kann, wird jedes Referenzattribut in einer anderen Farbe dargestellt und beim Berühren mit dem Mauszeiger wird der Name des Attributs und sein Wert in einem Popup-Label angezeigt. Ein Reaktionsoperator besitzt eine Hauptansicht, in die mit Drag&Drop Aktionen eingefügt werden können.

Eine neue Auswahlgruppe wird erstellt, indem per Drag&Drop ein neues Auswahlgruppenelement in die linke Seite eines Multimedia-Projektorelements eingefügt wird. In eine Auswahlgruppe können dann Präsentationsalternativen eingefügt werden, in die dann wiederum per Drag&Drop Verweise auf die in der Alternative enthaltenen Dokumentelemente eingetragen werden.

In Abbildung 4-4 zeigt das Fenster unten in der Mitte die Hauptansicht eines Multimedia-Projektorelements. Dieses Dokumentelement enthält ein Video *CometAnim*, das durch einen *while*-Operator mit einer Sprachsequenz verknüpft ist. Des weiteren ist ein Textelement *Heading* enthalten, das eine Überschrift repräsentiert und mit einem *while*-Operator mit dem umgebenden Multimedia-Projektorelement verknüpft ist. Der Knopf ist über einen Reaktionsoperator ebenfalls mit dem umgebenden Multimedia-Projektorelement verknüpft, und bewirkt beim Drücken das Beenden der Präsentation. Das Multimedia-Projektorelement enthält zudem ein Bild *CometPict*, das mittels eines *while*-Operators mit der Sprachsequenz verbunden ist. Da das Bild als alternative Präsentation anstelle des Videos eingesetzt werden können soll, ist auf der linken Seite der Hauptansicht eine Auswahlgruppe mit zwei Präsentationsalternativen eingefügt. Die erste Präsentationsalternative enthält das Video und hat die Priorität 40. Die zweite Präsentationsalternative mit der Priorität 10 enthält das Bild. Im Editor, wie auch später in der Präsentation, kann immer nur genau eine Präsentationsalternative einer Auswahlgruppe aktiv sein. Die aktive Präsentationsalternative in der Auswahlgruppe wird auf der linken Seite der Hauptansicht dunkel unterlegt. Auf der rechten Seite der Hauptansicht erscheint die Beschriftung inaktiver Elemente grau. Die aktive Präsentationsalternative bestimmt der Autor durch Anklicken.

4.5.1 Unterstützung bei der hierarchischen Strukturierung

Mit dem Tiempo-Editor können nur korrekte hierarchische Spezifikationen erzeugt werden. Dokumentelemente können nur so geschachtelt werden, wie es in einer Tiempo-DTD vorgesehen ist. Basierend auf einer Korrektheitsprüfung beim Einfügen neuer Dokumentelemente vervollständigt der Editor automatisch die hierarchische Schachtelung von Dokumentelementen. Um beispielsweise ein Fenster in ein Dokument einzufügen, das ein Video enthält, müßte zuerst ein Fenster-Dokumentelement in die Hauptansicht des Dokuments eingefügt werden, dann in dessen Hauptansicht ein Video-Projektorelement und in dessen Hauptansicht das Medienelement welches das entsprechende Video repräsentiert. Um diesen Vorgang abzukürzen, kann der Autor auch das Video-Medienelement per Drag&Drop in die Hauptansicht des Dokuments ziehen. Der Editor prüft nun, ob das Medienelement direkt im Dokument enthalten sein darf. Ist dies nicht der Fall, fügt der Editor automatisch die benötigte Dokumentelementhierarchie ein. Dieser Automatismus reduziert den Spezifikationsaufwand erheblich, kann jedoch auch dazu führen, daß nicht gewünschte Dokumentelemente eingefügt werden, wenn unterschiedliche Möglichkeiten zur Ergänzung einer Hierarchie bestehen. Dies kann der Autor korrigieren, indem er auf einer bestimmten Hierarchiestufe das unerwünschte Element austauscht. Hierzu muß er per Drag&Drop eine Instanz des gewünschten Elements auf das unerwünschte Element ziehen. Falls das neue Element äquivalent zum alten Element ist, d.h. dessen spezifizierten Inhalt und die spezifizierten Beziehungen zu anderen Elementen übernehmen kann, wird das alte Element durch das neue Element ersetzt.

4.5.2 Unterstützung bei der zeitlichen Strukturierung

Zur Vermeidung von Fehlern in der temporalen Dokumentspezifikation integriert der Editor einen Konsistenzprüfungsmechanismus, der auf den in Abschnitt 5 beschriebenen Algorithmen basiert. Der Konsistenzprüfungsmechanismus wird angestoßen, wenn der Autor ein neues Dokumentelement in eine Spezifikation einfügt bzw. entfernt sowie wenn er Attributwerte eines Elements verändert. Falls die Aktion zu einer inkonsistenten temporalen Spezifikation führt, wird der Autor darauf hingewiesen. Führt die Aktion zu keiner Inkonsistenz, wird für temporale Attribute, deren Wert nicht vom Autor spezifiziert wurde, der maximale Dienstgütebereich anhand der Abhängigkeiten zu spezifizierten Attributen bestimmt. Dadurch hat der Autor stets Kenntnis von den korrekten Werten aller Attribute, wodurch lästige manuelle Rechenarbeit bei

der Bestimmung des zeitlichen Layouts verhindert wird. Hält sich der Autor zudem an die angezeigten maximalen Dienstgütebereiche, bleibt die Dokumentspezifikation fehlerfrei.

Der Editor unterstützt zwei Spezifikationsmodi: Einen Modus in dem die temporale Konsistenzprüfung bei jeder Aktion des Autors durchgeführt wird und einen Modus in dem keine temporale Konsistenzprüfung stattfindet (Die Konsistenzprüfung hinsichtlich der Schachtelung von Dokumentelementen wird immer durchgeführt). Der Autor kann zwischen diesen Modi beliebig hin- und herschalten. Der Modus ohne Konsistenzprüfung ist in der Entwurfsphase sinnvoll, in der die temporale Struktur des Dokuments noch im Fluß ist.

4.5.3 Simulation von Adaptionen

In der **zeitlichen Ansicht** (Temporal View) eines Dokuments wird das temporale Layout des Dokuments im Detail dargestellt. Die zeitliche Ansicht folgt dem Zeitachsen-basierten Visualisierungsansatz. Die durch Projektoren implizierte Präsentation von Medien wird durch Balken repräsentiert, deren Länge der Präsentationsdauer des Mediums entspricht und deren Höhe die Präsentationsgeschwindigkeit repräsentiert. Balken werden in der zeitlichen Ansicht so angeordnet, wie es durch Intervalloperatoren definiert wurde. In der zeitlichen Ansicht wird immer die gerade aktive Präsentationsalternative jeder Auswahlgruppe dargestellt.

In der zeitlichen Ansicht existieren Schieberegler mit denen man eine verfügbare Bandbreite, eine verfügbare Speichermenge und eine verfügbare Menge CPU-Zyklen zwischen dem minimalen und maximalen Ressourcenbedarf der spezifizierten Präsentation einstellen kann. Im Fenster wird dann genau die zeitliche Anordnung von Medienelementen dargestellt, die optimal für diese Menge Ressourcen ist. Zur Berechnung der Anordnung wird der in Abschnitt 8 beschriebene Ablaufplanungsalgorithmus eingesetzt. Mit Hilfe solcher Simulationen können Dienstgütebereiche im Detail bestimmt werden oder der Nutzen von Auswahlgruppen überprüft werden. Abbildung 4-5 zeigt die zeitliche Ansicht des Dokuments aus Abbildung 4-4.

4.6 Layout-Definition

Jedes Dokumentelement der Tiempo-Sprache mit einer abgeschlossenen räumlichen Ausprägung besitzt eine **räumliche Ansicht** (Spatial View). Dies sind Dokumentelemente mit denen

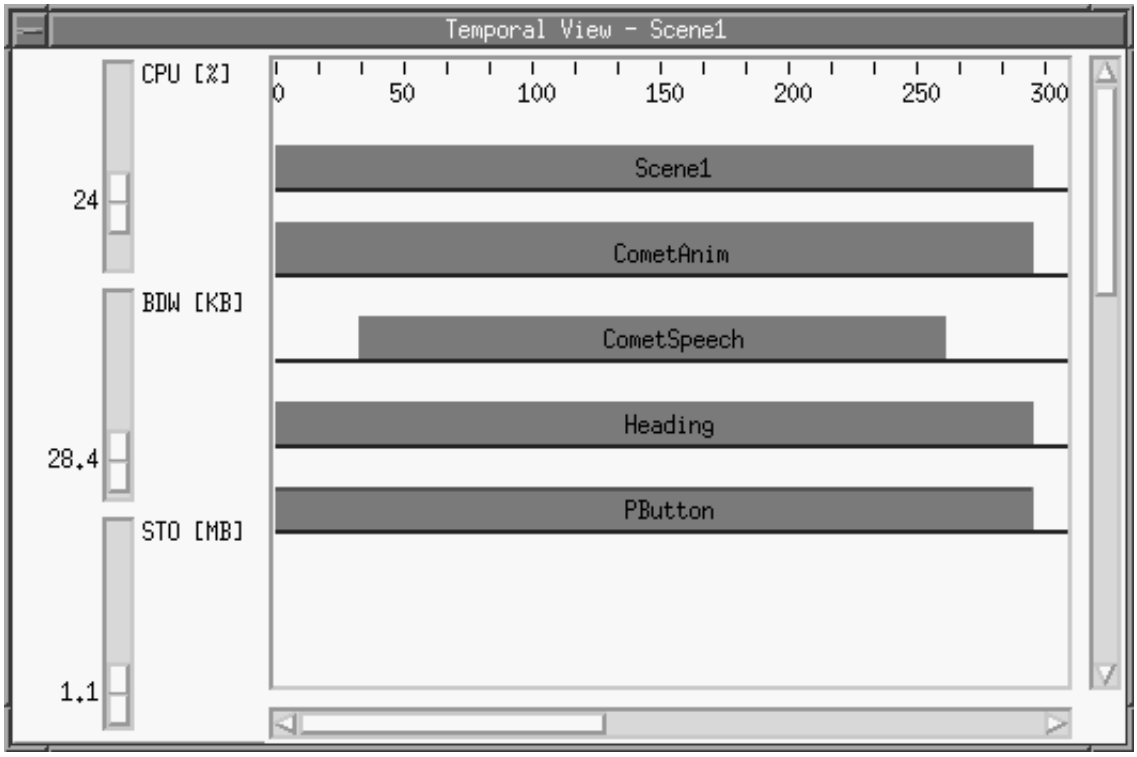


Abb. 4-5 : Überprüfung der zeitlichen Strukturierung

der Bildschirm oder Fenster assoziiert werden. Die räumliche Ansicht eines Dokumentelements folgt dem Bildschirm-basierten Spezifikationsansatz. In der räumlichen Ansicht wird die durch Projektoren implizierte Präsentation von Medien so dargestellt, wie sie in einer Präsentation sein wird (WYSIWYG-Ansatz). Da im Laufe der Präsentation unterschiedliche Medien an derselben Stelle dargestellt werden können, wird in der räumlichen Ansicht nur ein Präsentationszeitpunkt dargestellt, der vorher in der zeitlichen Ansicht bestimmt werden muß. In Abbildung 4-6 ist die räumliche Ansicht des zusammengesetzten Elements *Scene1* aus Abbildung 4-4 zu sehen.

In der räumlichen Ansicht kann die Position der Medien durch Verschieben mit der Maus bestimmt werden. Auch in der räumlichen Ansicht hat jedes Element ein Kontextmenü mittels dem seine Darstellung verändert und die Überlappung mit anderen Elementen bestimmt werden kann. Um dem Autor die Identifikation zu erleichtern, wird beim Berühren eines Mediums mit der Maus ein Popup-Label mit dem Namen des Projektors sichtbar.

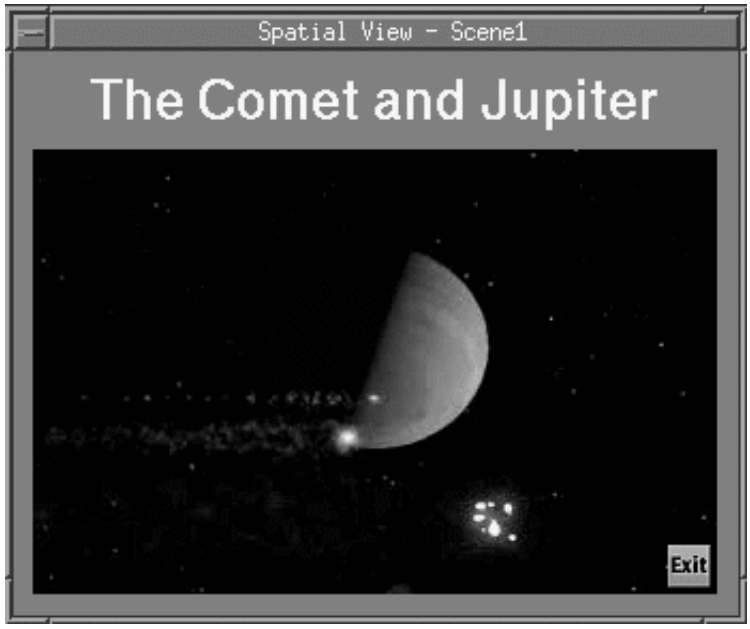


Abb. 4-6 : Bestimmung des räumlichen Layouts

4.7 Testen von Dokumentspezifikationen

Im Tiempo-Dokumentensystem ist der Tiempo-Editor mit dem Tiempo-Präsentationssystem integriert. Dadurch können während des Spezifikationsprozesses erstellte Dokumente teilweise oder komplett präsentiert werden.

Um die Präsentation bei bestimmten Ressourcensituationen zu simulieren, kann eine bestimmte Ressourcenmenge in der zeitlichen Ansicht eingestellt werden. Hierbei werden dann die geeigneten Präsentationsalternativen gewählt. Es können jedoch auch nur bestimmte Kombination von Präsentationsalternativen der Auswahlgruppen getestet werden. Bei allen Tests wird in der zeitlichen Ansicht dargestellt, wie die Präsentation ablaufen wird.

4.8 Zusammenfassung

Der Tiempo-Editor visualisiert eine Dokumentspezifikation durch drei Ansichten. Die Hauptansicht folgt dem Flowchart-basierten Visualisierungsansatz und ermöglicht so eine abstrakte Darstellung der Dokumentstruktur. Die auf dem Zeitachsen-basierten Ansatz basierende zeitliche Ansicht stellt den temporalen Ablauf eines Dokuments dar. Mit Hilfe der räumlichen Ansicht, die einen Bildschirm-basierten Spezifikationsansatz realisiert, wird das räumliche

Layout eines Dokuments bestimmt. Durch die Kombination der drei Visualisierungsansätze wird die Stärke jedes Ansatzes optimal genutzt, wodurch eine übersichtliche Darstellung der unterschiedlichen Aspekte einer Dokumentspezifikation möglich wird. Objektorientierte Interaktionsmechanismen ermöglichen die einfache Komposition von Dokumenten. Das Autorensystem von Firefly, das ebenfalls flexible Spezifikationen unterstützt, kennt nur eine temporale Ansicht, wodurch insbesondere bei komplexeren Dokumenten die Übersichtlichkeit nicht immer gegeben ist und das räumliche Layout ohne Visualisierung spezifiziert werden muß. Im Vergleich dazu erlaubt das Autorensystem von Mbuild die graphisch-interaktive Spezifikation des zeitlichen und räumlichen Layouts. Das räumliche Layout muß jedoch mit abstrakten Objekten ohne WYSIWYG-Darstellung erstellt werden.

Um der höheren strukturellen Komplexität adaptiver Dokumente zu begegnen, integriert der Tiempo-Editor Konsistenzprüfungsmechanismen zur Fehlervermeidung und es können Simulationen durchgeführt werden, um den Nutzen von Präsentationsalternativen zu verifizieren. Firefly, Mbuild und MODE bieten solche Möglichkeiten nicht an.

Durch die Integration des Tiempo-Editors und des Tiempo-Präsentationssystems können während des Spezifikationsprozesses Dokumentteile auf einfache Weise präsentiert werden, wodurch ohne Aufwand überprüft werden kann, ob das Dokument den Wünschen des Autors entspricht.

5 Konsistenzprüfung adaptiver Multimedia-Dokumente

Die in Kapitel 3 vorgestellte Dokumentensprache erlaubt die Spezifikation interaktiver adaptiver Multimedia-Dokumente. Damit eine Multimedia-Präsentation erzeugt werden kann, muß die Dokument-Spezifikation konsistent sein.

Um die Konsistenz einer Dokument-Spezifikation zu verifizieren, kann man das Dokument präsentieren und falls Fehler auftreten, die Spezifikation korrigieren. Dieser Ansatz des manuellen Testens hat jedoch einen gravierenden Nachteil. Für komplexe Dokumente mit vielen Interaktionen ist es nahezu unmöglich mit vertretbaren Aufwand alle möglichen Interaktionspfade zu testen. Deshalb sind selbst in professionellen Multimedia-Dokumenten häufig Fehler enthalten, wie beispielsweise Knöpfe deren Betätigung nichts bewirkt. Für adaptive Dokumente ist das manuelle Testen noch viel schwieriger, da das Überprüfen aller möglichen Kombinationen von Präsentationsalternativen das Erzeugen der entsprechenden Ressourcensituationen erfordert. Dies ist jedoch sehr aufwendig, wenn nicht sogar unmöglich. Die Lösung für diese Problematik ist ein automatisches Testen von Dokument-Spezifikationen durch ein Werkzeug, genannt **Konsistenzprüfer**, der alle möglichen Präsentationssituationen durchspielt und so feststellt, ob Fehler in der Spezifikation enthalten sind.

In Abschnitt 5.1 wird die Konsistenz von Multimedia-Dokumenten näher betrachtet. Dann wird ein Konsistenzprüfungsverfahren für das Tiempo-Modell vorgestellt.

5.1 Konsistenzaspekte

Die Konsistenz eines Dokuments kann auf Dokumentelementebene und Attributebene betrachtet werden. Der in Kapitel 4 vorgestellte Tiempo-Editor integriert einen Konsistenzprüfer, der entsprechende Konsistenzüberprüfungen durchführt.

5.1.1 Konsistenz auf Dokumentelementebene

Ein Dokument ist konsistent auf Dokumentelementebene, wenn die Dokumentelemente die erforderlichen Attribute besitzen und ihre (hierarchische) Strukturierung korrekt ist. Damit der Konsistenzprüfer im Tiempo-Editor unterschiedliche Tiempo-Dialekte verarbeiten kann, wer-

den die jeweiligen Regeln für ein korrekt strukturiertes Dokument dynamisch anhand der beim Start geladenen DTD konfiguriert.

5.1.2 Konsistenz auf Attributebene

Die Konsistenz auf Attributebene erfordert, daß die spezifizierten Attributwerte mit den Attributtypen konform sind. Darüber hinaus müssen entsprechend dem Tiempo-Modell bestimmte Abhängigkeiten von Attributen, die das zeitliche und räumliche Layout definieren, eingehalten werden.

Da die Datenräume von Medienobjekten horizontal und vertikal relativ zum umgebenden zusammengesetzten Datenraum, jedoch aus Sicht des Datenraums absolut positioniert werden, sind entsprechende Attribute unabhängig voneinander. Ebenso ist es erlaubt, daß in einem zusammengesetzten Datenraum enthaltene Datenräume räumlich aus diesem herausragen können. Entsprechende Teile werden einfach nicht präsentiert. Hinsichtlich der horizontalen und vertikalen Positionierung ist damit keine Konsistenzprüfung erforderlich.

Bei den Attributen, welche die räumliche Überlappung von Datenräumen definieren, ist undefiniert was passiert, wenn sich überlappende Datenräume in derselben Ebene befinden. Daher ist eine Dokumentspezifikation nur konsistent, wenn sich alle Datenräume in einem zusammengesetzten Datenraum in unterschiedlichen Ebenen befinden.

Die spezifizierten zeitlichen Attributwerte von Intervalloperatoren, Projektoren und Auswahlgruppen müssen ein konsistentes zeitliches Layout ergeben. Hierzu definieren wir:

Def.: Zeitliche Konsistenz

> Eine Dokumentspezifikation ist zeitlich konsistent, wenn es für alle zeitlichen Attribute Werte bzw. mindestens eine Belegung innerhalb der spezifizierten Dienstgütebereiche gibt, so daß der zugehörige zeitliche Ablauf der Präsentation widerspruchsfrei ist.

Die Betonung bei dieser Definition liegt auf mindestens einer Belegung. Es darf durchaus Werte in Dienstgütebereichen geben, die zu widersprüchlichen zeitlichen Anordnungen führen können. Es muß jedoch mindestens eine ohne Widersprüche und damit konsistente Anordnung

geben. Im folgenden wird ein **Konsistenzprüfungsverfahren** zur Validierung der zeitlichen Konsistenz von Tiempo-Dokumenten vorgestellt.

Aufgrund der flexiblen Spezifikationen, die das Tiempo-Modell ermöglicht, ist die potentielle Verletzung der zeitlichen Konsistenz durch Spezifikationsaktionen nicht unmittelbar vom Autor zu erkennen. Dieses Problem wird im Tiempo-Editor durch ein **Konsistenzsicherungsverfahren** beseitigt, das auf dem Begriff der vollständigen zeitliche Konsistenz basiert:

Def.: Vollständige zeitliche Konsistenz

Eine Dokumentspezifikation ist vollständig zeitlich konsistent, wenn es für alle spezifizierten singulären Werte und alle Werte innerhalb der spezifizierten Dienstgütebereiche von zeitlichen Attributen einen widerspruchsfreien zeitlichen Ablauf gibt.

Diese Definition ist wesentlich strikter als die der zeitlichen Konsistenz, da nun kein Dienstgütebereich auch nur einen Wert enthalten darf, der eine Inkonsistenz verursacht.

Das Konsistenzsicherungsverfahren ermöglicht es, daß einem Autor für alle zeitlichen Attribute der vollständig konsistente Wertebereich angezeigt wird. Das ist der Wertebereich, welcher das Kriterium der vollständigen zeitlichen Konsistenz erfüllt. Wählt der Autor bei der Spezifikation von Attributwerten nur Werte aus diesen Wertebereichen, entstehen fehlerfreie Dokumente. Die algorithmischen Aspekte des Konsistenzsicherungsverfahrens werden in den nächsten Abschnitten erläutert.

5.2 Konsistenzprüfungsmodell Ereignisgraph

Da sich zeitliche Abhängigkeiten gut durch Graphen darstellen lassen, verwenden wir als Konsistenzprüfungsmodell für die zeitliche Konsistenzprüfung gerichtete azyklische Graphen, sogenannte **Ereignisgraphen**. Die Knoten in einem Ereignisgraph repräsentieren Ereignisse, wie den Anfang oder das Ende einer Medienpräsentation. Die Kanten beschreiben die zeitlichen Abstände zwischen den Ereignissen.

5.2.1 Modellierung von Dokumentelementen

Präsentations- und Interaktionsintervalle von Projektorelementen sind in Ereignisgraphen durch einen Anfangs- und einen Endknoten repräsentiert, die durch eine gerichtete Kante verbunden sind, deren Länge der Länge des jeweiligen Intervalls entspricht. Datenräume werden ebenfalls durch einen Anfangs- und einen Endknoten repräsentiert, die durch eine Kante verbunden sind, welche die Länge ihrer t-Achse beschreibt. Projektionen sind durch geeignete Kanten zwischen den Präsentationsintervallknoten und den Datenraumknoten modelliert. Abbildung 5-1 zeigt, welche Kanten in Abhängigkeit des Projektionstyps eingeführt werden müssen. Beispielsweise muß bei der spezifizierten Ausrichtungspolitik *AlignLength* eine Kante der Länge Null zwischen dem Anfangsknoten des Präsentationsintervalls und des Datenraums eingezogen werden sowie eine Kante der Länge Null zwischen dem Endknoten des Datenraums und des Präsentationsintervalls. Intervalloperatoren werden in Abhängigkeit ihres Typs durch entsprechende Kanten zwischen den Präsentationsintervall-, Interaktionsintervall- oder Datenraumknoten repräsentiert. Die in Interaktionsmanagern definierten Abstände werden durch Kanten zwischen Präsentationsintervall- und Interaktionsintervallknoten modelliert.

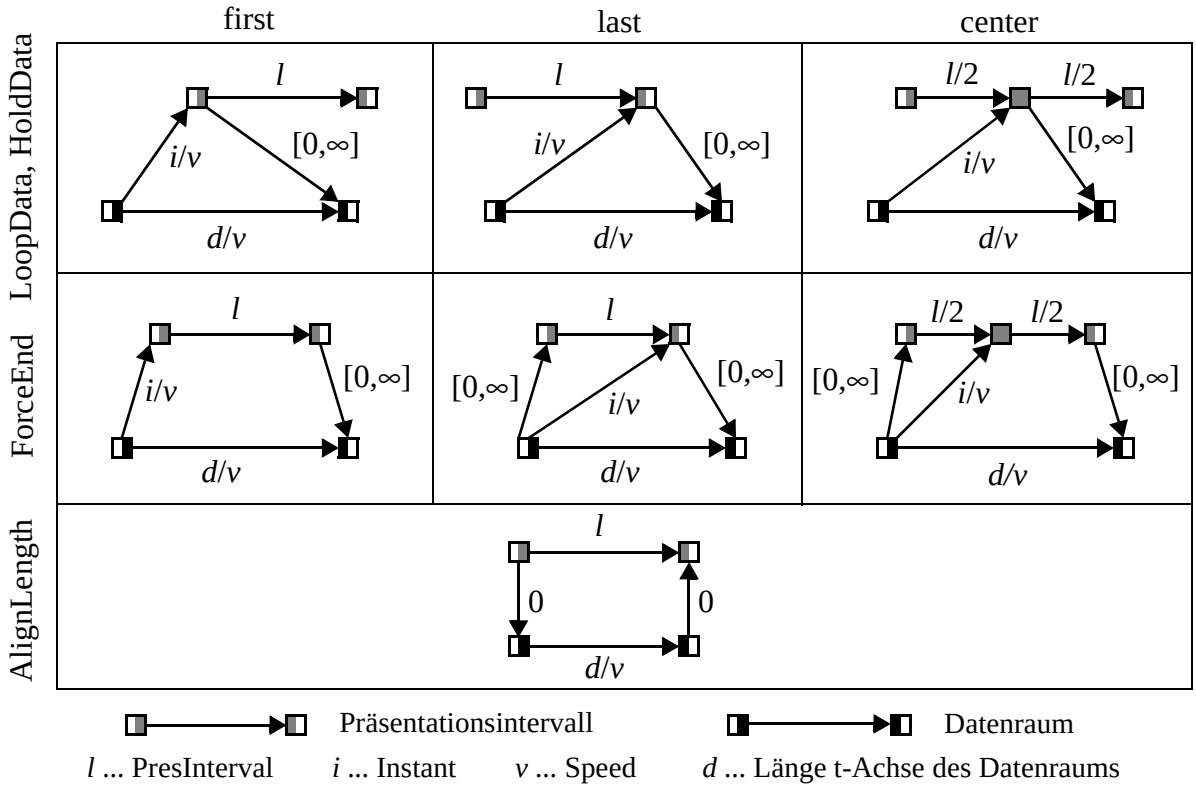


Abb. 5-1 : Modellierung von Projektionen in Ereignisgraphen

5.2.2 Bestimmung von Kantenlängen

Da das Tiempo-Modell flexible zeitliche Spezifikationen ermöglicht, können Kanten variable Längen haben. Die Kanten in einem Ereignisgraph werden somit entweder mit einem einzelnen Längenwert oder einem Längenintervall markiert. Die effektive Länge l_k einer Kante k muß aus der Sicht des äußersten Datenraums T_h, der im Graph repräsentiert ist, angegeben werden. Das bedeutet, der im Dokumentelement spezifizierte Längenwert $\underset{\sim}{l}_k$ muß mit den spezifizierten Präsentationsgeschwindigkeiten $\underset{\sim}{v}^i$ aller geschachtelten Datenräume dividiert werden, in denen das Dokumentelement direkt oder indirekt enthalten ist, um die Länge der entsprechenden Kante im Ereignisgraph zu bestimmen:

$$l_k = \underset{\sim}{l}_k \Big/ \prod_{i=q}^{h} \underset{\sim}{v}^i \qquad \text{falls} \quad k \in T_q \in T_{q+1} \in \ldots \in T_{h-1} \in T_h \tag{5-1}$$

Abbildung 5-1 zeigt, wie Kanten, die Projektionen modellieren, diesbezüglich zu behandeln sind. Der Divisor in Formel (5-1) beschreibt die effektive Präsentationsgeschwindigkeit v^q des Datenraums T_q bzw. des Projektors q.

5.2.3 Generierung von Ereignisgraphen

Ein Tiempo-Dokument besteht aus ineinander projizierten Datenräumen. Aufgrund der Projektionen kann man hierbei auch von geschachtelten Datenräumen sprechen. Der Ereignisgraph für ein Dokument wird entsprechend der hierarchischen Schachtelung der Datenräume bzw. Projektorelemente von innen nach außen aufgebaut. Es werden zuerst Teilereignisgraphen für die in einem Multimedia-Projektor enthaltenen Projektorelemente gebildet, welche entsprechend der spezifizierten Intervalloperatoren durch Kanten verknüpft werden. Dann werden die Knoten und Kanten hinzugefügt, welche Präsentationsintervall, Datenraum und Projektion des Multimedia-Projektors repräsentieren, und die Präsentationsintervallknoten der Teilereignisgraphen enthaltener Projektoren mit den Datenraumknoten des Multimedia-Projektors verknüpft. Hierbei ist folgendes zu beachten. Zusammengesetzte Datenräume können mit unterschiedlichen Datenräumen beginnen bzw. enden, dürfen aber laut dem Tiempo-Modell nicht länger sein als ihr Inhalt. Solche Datenräume bezeichnet man als **indeterministische Datenräume** und die mit ihnen assoziierten Projektorelemente als **indeterministische Projektoren**. Diese Form von Indeterminismus kann in einem einzelnen Ereignisgraph nicht geeignet modelliert werden.

Daher muß für jede mögliche Anfang-Ende-Kombination eines Datenraums ein separater Ereignisgraph erstellt werden.

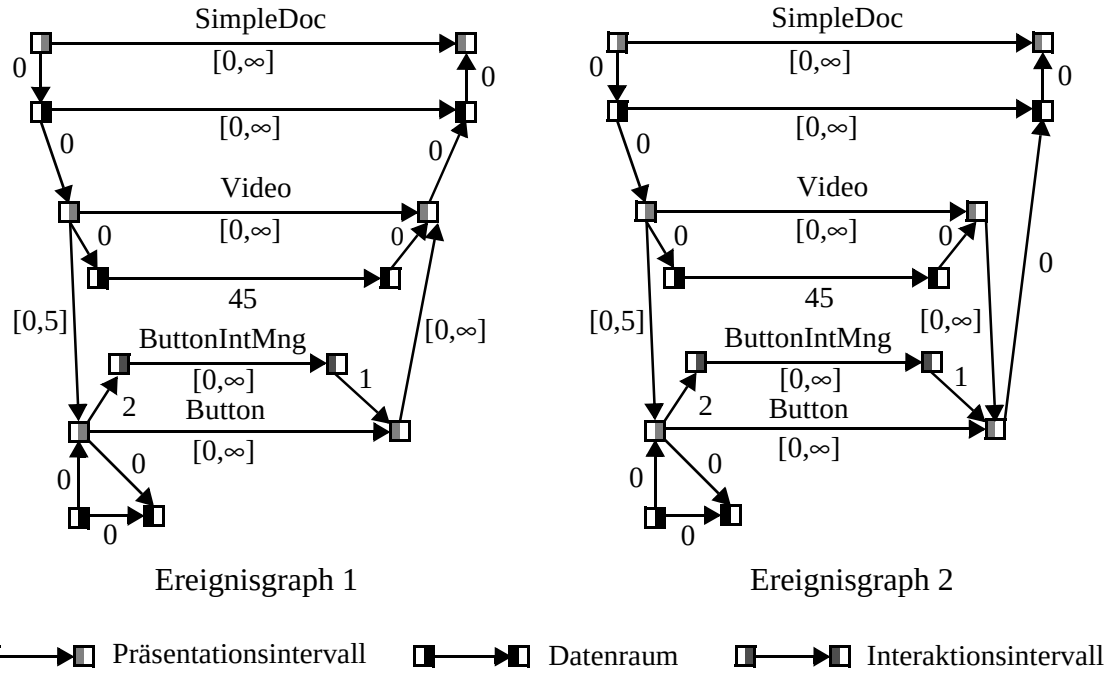
Hierzu bestimmt man die Teilgraphen für die enthaltenen Projektoren und Intervalloperatoren. Es kann mehrere Teilgraphen geben, da die in einem zusammengesetzten Medium enthaltene zeitliche Spezifikation nicht zusammenhängend sein muß. Für jeden der Teilgraphen bildet man nun die Kombinationen (*A*, *B*) aller Knoten *A* und *B*, für die gilt: *A* ist Anfangsknoten eines Präsentationsintervalls und hat keine eingehenden Kanten. *B* ist Endknoten eines Präsentationsintervalls und hat keine ausgehenden Kanten. Bildet man die Menge aller Kombinationen (*A*, *B*), erhält man alle möglichen Anfang-Ende-Kombinationen des zusammengesetzten Datenraums.

Für jede Anfang-Ende-Kombination (*A*, *B*) wird ein separater Ereignisgraph wie folgt gebildet. Zwischen Knoten *A* und allen anderen Knoten ohne eingehende Kanten wird jeweils eine Kante der Länge [0, ∞] eingefügt und zwischen allen Knoten ohne ausgehende Kanten und dem Knoten *B* wird jeweils eine Kante der Länge [0, ∞] eingefügt. Ist der Knoten *A* nicht der Anfangsknoten des zusammengesetzten Datenraums, wird zwischen dem Anfangsknoten des Datenraums und dem Knoten *A* ein Kante der Länge Null eingefügt. Ist der Knoten *B* nicht der Endknoten des zusammengesetzten Datenraums, wird zwischen dem Knoten *B* und dem Endknoten des Datenraums eine Kante der Länge Null eingefügt. Ein auf diese Weise erzeugter Ereignisgraph beginnt und endet eindeutig, d.h. mit einem bestimmten Intervalloperator oder Projektorelement.

Teilweise kann man, wie noch gezeigt werden wird, durch den geschickten Einsatz von Algorithmen im Rahmen der Konsistenzprüfung vermeiden, daß Ereignisgraphen für alle möglichen Anfang-Ende-Kombinationen gebildet werden müssen.

Um die Generierung eines Ereignisgraphen zu verdeutlichen, betrachten wir nun das Beispiel in Abbildung 5-2. Im oberen Teil der Abbildung ist eine Dokumentspezifikation zu sehen und im unteren Teil die Ereignisgraphen für die Dokumentspezifikation. Es entstehen zwei Ereignisgraphen, da zwischen dem Video und dem Knopf ein *Cobegin*-Operator spezifiziert ist, wodurch der Datenraum des *SimpleDoc* entweder mit dem Video oder dem Knopf enden kann. Der Datenraums des Videos ist nur 45 Zeiteinheiten lang, da das Video mit doppelter Geschwindigkeit ausgespielt werden soll. Wie zu sehen ist, ist die Repräsentation des Datenraums des

```
<Document  Id="SimpleDoc">
        <Projection  PresInterval="[0:0,0:?]" Speed="1.0" Rate="1.0" >
                <AlignLength/>
        </Projection>
        <VideoPro  Id="Video">
                <Projection  PresInterval="[0:0,0:?]" Speed="2.0" Rate="1.0" >
                        <AlignLength/>
                </Projection>
                <Video  Length="90" .../>
        </VideoPro>
        <ButtonPro  Id="Button">
                <Projection  PresInterval="[0:0,0:?]" Speed="0" Rate="1.0" >
                        <HoldData  Instant="0" Position="first" />
                </Projection>
                <ButtonIntMng  Id="BIM" IntInterval="[0:0,0:?]" StartDelay="2" EndDelay="1" .../>
                <Picture .../>
        </ButtonPro>
        <Cobegin  From="Video" To="Button" Delay1="[0:10,5:0]"/>
</Document>
```



Ereignisgraph 1 Ereignisgraph 2

Abb. 5-2 : Ereignisgraph Beispiel

Knopfs unnötig, da sie keine relevanten Informationen beinhaltet. Der Datenraum diskreter Medien wird in Ereignisgraphen, die zur Konsistenzprüfung eingesetzt werden, daher nicht repräsentiert.

5.2.4 Konsistenzprüfung und Längenbestimmung in Ereignisgraphen

Um die zeitliche Konsistenz eines Ereignisgraphen zu prüfen, muß festgestellt werden, ob es keine Widersprüche hinsichtlich der Abstände zwischen Knoten gibt. Des weiteren sollen die Kantenlängen ermittelt werden, für welche die vollständige Konsistenz gegeben ist. Hierzu wird der folgende **Längenbestimmungsalgorithmus** angewendet.

Die Längenbestimmung basiert auf dem in [DMP89] beschriebenen Algorithmus: Ein Ereignisgraph wird als Adjazenzmatrix dargestellt. Hierbei wird jede Kante, die von einem Knoten *A* zu einem Knoten *B* geht und die Länge [*l*, *u*] hat, durch zwei Einträge in der Adjazenzmatrix repräsentiert. In die Zelle [*A*, *B*] wird die maximale Länge *u* einer Kante eingetragen. In die Zelle [*B*, *A*] wird die minimale Länge der Kante *l* mit einem negativen Vorzeichen eingetragen. Hat eine Kante eine fixe Länge *l* wird in Zelle [*A*, *B*] mit *l* und Zelle [*B*, *A*] mit -*l* belegt. Existiert für ein Knotenpaar keine Kante im Ereignisgraph trägt man in Zelle [*A*, *B*] ∞ und in Zelle [*B*, *A*] -∞ ein.

Auf die Adjazenzmatrix wird nun der Algorithmus von Floyd-Warshall [Sedg88] angewendet. Dieser Algorithmus dient bei herkömmlichen Graphen zur Berechnung der minimalen Abstände zwischen allen Knoten des Graphen. Angewendet auf eine derart konstruierte Adjazenzmatrix, wird jedoch folgendes berechnet:

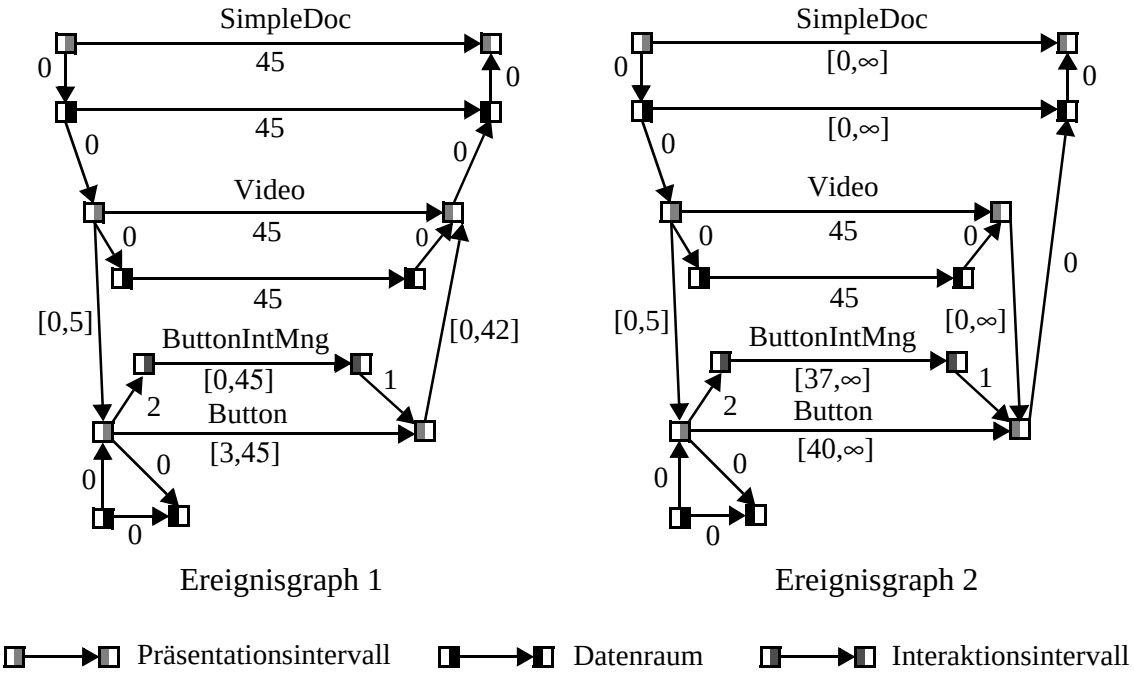
- Enthält die berechnete Matrix in der Diagonalen negative Werte, ist der repräsentierte Graph nicht konsistent. Mit anderen Worten, es gibt keine Kombination von Kantenwerten im Graphen, so daß alle minimalen und maximalen Abstände eingehalten werden.

- Enthält die berechnete Matrix in der Diagonalen nur Nullen, ist der Graph konsistent. D.h. es gibt mindestens eine Kombination von Kantenwerten, so daß alle minimalen und maximalen Abstände zwischen Knoten eingehalten werden können. Hierbei enthält die berechnete Matrix die möglichen minimalen und maximalen Abstände zwischen allen Knoten.

Im zweiten Fall kann man aus der berechneten Adjazenzmatrix, der sogenannten **Längenmatrix**, nun die vollständig konsistente minimale und maximale Länge jeder Kante aus den gleichen Zellen auslesen, in die man vorher die spezifizierten minimalen und maximalen Längen eingetragen hat. Wenn man die Werte aus der Längenmatrix in einen Ereignisgraph einträgt, erhält man einen **vollständig konsistenten Ereignisgraph**. Darüber hinaus stehen auch in Zel-

len, für die es keine Kante im Ereignisgraph gibt, die minimalen und maximalen Abstände der entsprechenden Knoten. Ist der minimale und maximale Abstand zwischen zwei Knoten ∞, befinden sich die Knoten nicht im gleichen Teilgraph, d.h. der Graph ist partitioniert.

Abbildung 5-3 zeigt die mit dem Längenbestimmungsalgorithmus berechneten vollständig konsistenten Ereignisgraphen aus Abbildung 5-2. Im Ereignisgraph 2 endet der Datenraum *SimpleDoc* mit der Präsentation des Knopfs. Da dessen Präsentationsintervall unendlich lang sein kann, kann auch die gesamte Dokumentpräsentation unendlich lang sein. Im Ereignisgraph 1 endet der umgebende Datenraum mit dem Ende der Videopräsentation, deren Länge fest vorgegeben war, und somit endet auch die Präsentation des Knopfs spätestens nach 45 Zeiteinheiten.



Ereignisgraph 1 Ereignisgraph 2

Abb. 5-3 : Vollständig konsistente Ereignisgraphen

Die Komplexität des Längenbestimmungsalgorithmuses wird durch den Floyd-Warshall-Algorithmus bestimmt. Dementsprechend besitzt der Längenbestimmungsalgorithmus eine Komplexität von $O(n^3)$ für einen Graph mit n Knoten.

5.3 Konsistenzverfahren für hierarchische Dokumente

Für größere Dokumente entstehen schnell große und eventuell auch viele Ereignisgraphen, deren Überprüfung mittels des Längenbestimmungsalgorithmuses aufgrund dessen kubischer Komplexität lange Dauern kann. Zur Effizienzsteigerung ist es daher sinnvoll, die hierarchische Struktur der Dokumente auszunützen und die Konsistenz für zeitlich abgeschlossene Dokumentteile mittels des Längenbestimmungsalgorithmuses parallel zu prüfen.

5.3.1 Bestimmung zeitlich abgeschlossener Datenräume

Die zeitliche Abgeschlossenheit ist wie folgt definiert:

Def.: Ein Datenraum *T* ist **zeitlich abgeschlossen**, wenn in *T* oder in den in *T* enthaltenen Datenräumen keine Querkanten beginnen oder enden, die aus *T* herausführen.

Querkanten repräsentieren über die Grenzen eines Datenraums hinausgehende zeitliche Abhängigkeiten. Diese entstehen, wenn Intervalloperatoren spezifiziert wurden, die Dokumentelemente in unterschiedlichen Datenräumen in Beziehung setzen. Abbildung 5-4 zeigt die schematische Darstellung eines Dokuments mit geschachtelten Datenräumen. In diesem Dokument verknüpft die Querkante *A* ein Medium im Datenraum T_1 mit einem Medium in T_6. Querkante *B* beschreibt eine Beziehung zwischen einem Medium in T_2 und T_3. Die Querkante *C* beschreibt eine Beziehung zwischen einem Medium in T_4 und T_6.

Die zeitlich abgeschlossenen Datenräume eines Dokuments können wie folgt bestimmt werden. Es sei *Q* die Menge aller Querkanten. Man bildet nun für jede Querkante *q* in *Q* die Menge T_q der Datenräume, die durchquert werden müssen, um – bei Berücksichtigung der hierarchischen Schachtelung des Dokuments – vom Anfangsknoten der Querkante bis zu ihrem Endknoten zu gelangen. Sind all Mengen T_q gebildet, werden die Mengen vereinigt, welche gleiche Datenräume beinhalten. Jede der auf diese Weise entstandenen Mengen enthält nun genau die Datenräume, welche gemeinsam betrachtet werden müssen. Die Menge der vereinigten Mengen zeitlicher Räume soll mit *V* bezeichnet werden. Der Datenraum einer vereinigten Menge, der alle anderen umschließt, ist zeitlich abgeschlossen. Datenräume, die in keiner der Mengen vorkommen sind ebenfalls zeitlich abgeschlossen. Für das Beispiel in Abbildung 5-4 wäre $Q = \{A, B,$

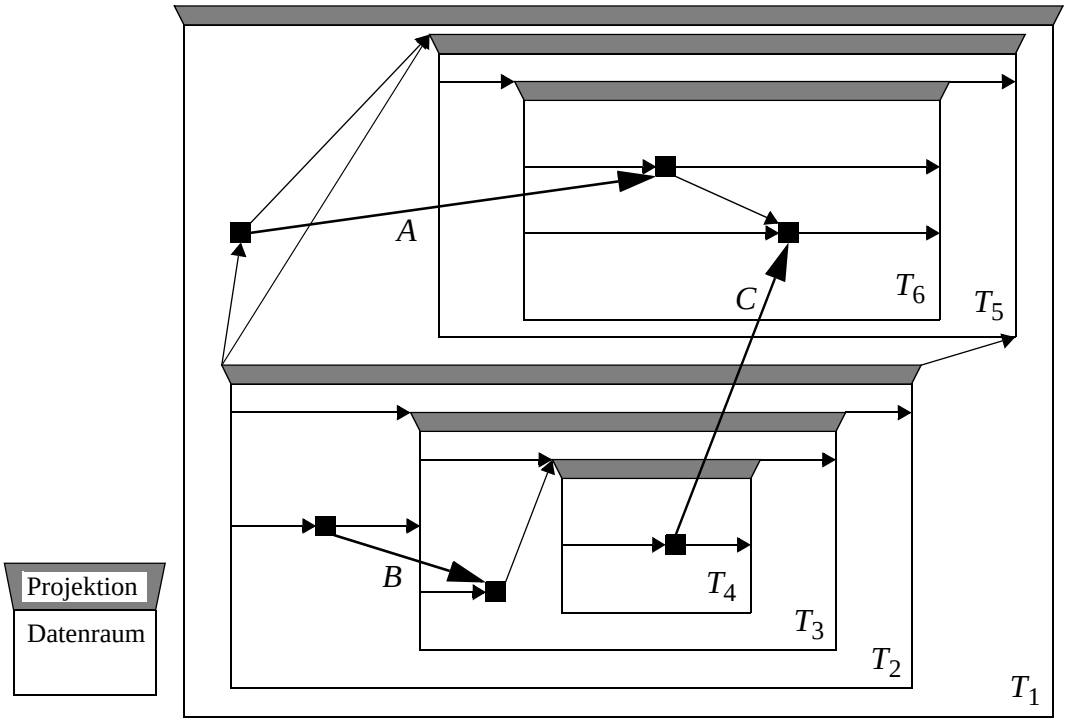


Abb. 5-4 : Konsistenzprüfung von hierarchischen Dokumenten

$C\}$, $T_A = \{T_1, T_5, T_6\}$, $T_B = \{T_2, T_3\}$ und $T_C = \{T_4, T_3, T_2, T_1, T_5, T_6\}$. Nach der Vereinigung erhält man die Menge $V = \{\{T_4, T_3, T_2, T_1, T_5, T_6\}\}$. Somit ist nur T_1 zeitlich abgeschlossen und die gesamte Spezifikation muß auf einmal auf Konsistenz geprüft werden. Würde im Beispiel die Querkante C fehlen, wäre $V = \{\{T_1, T_5, T_6\}, \{T_2, T_3\}\}$ und man hätte die zeitlich abgeschlossenen Räume T_1, T_2, und T_4, deren Konsistenz separat überprüft werden könnte.

5.3.2 Konsistenzverfahren für zeitlich abgeschlossene Datenräume

Sind zeitlich abgeschlossene Datenräume hierarchisch geschachtelt, läßt sich ihre Konsistenz effizient überprüfen. Abbildung 5-5 zeigt schematisch die Schachtelung der Datenräume eines Dokuments, die alle zeitlich abgeschlossen sind. Das Konsistenzverfahren wird entsprechend dieser Schachtelung in zwei Schritten durchgeführt:

1. **Konsistenzprüfung:** Die Konsistenz der Spezifikation wird von innen nach außen entsprechend der Schachtelung der Datenräume geprüft. In Abbildung 5-5 könnte dies beispielsweise in der Reihenfolge T_4, T_3, T_5, T_7, T_8, T_6, T_2, T_1 passieren. Für jedes Projektorelement wird zuerst die Konsistenz seines Datenraums überprüft und dann die Konsistenz seiner

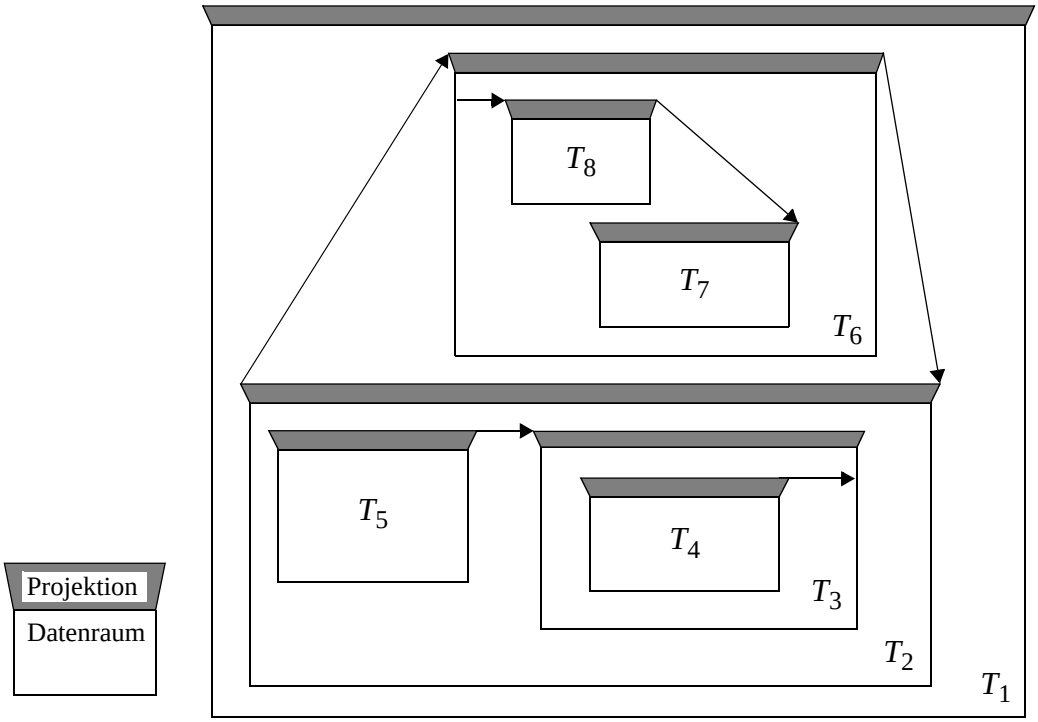


Abb. 5-5 : Konsistenzprüfung zeitlich abgeschlossener Dokumentteile

Projektion. Dabei wird die Länge seines Präsentationsintervalls berechnet. Mit Hilfe der Präsentationsintervallänge kann dann auf der nächsthöheren Stufe die Konsistenz dieses Datenraums geprüft werden.

Die Länge des Datenraums eines einfachen Mediums ist fix und im Dokument enthalten, dementsprechend ist nichts zu prüfen. Der Datenraum eines zusammengesetzten Mediums ist durch die enthaltene Spezifikation bestimmt. Um seine Konsistenz zu prüfen, erstellt man einen Ereignisgraph aus den darin enthaltenen Projektoren und Intervalloperatoren und wendet den Längenbestimmungsalgorithmus an. Ist der Graph konsistent, sucht man aus der Längenmatrix das Maximum der maximalen Kantenlängen kleiner unendlich und das Maximum der minimalen Kantenlängen kleiner unendlich heraus und erhält damit die minimal und maximal mögliche Länge des zusammengesetzten Datenraums.

Um die Projektion eines Datenraums zu prüfen, wird lediglich ein Ereignisgraph, wie er in Abbildung 5-1 zu sehen ist, erstellt und darauf der Längenbestimmungsalgorithmus angewendet. Ist die Projektion konsistent, kann aus der Längenmatrix die Präsentationsintervallänge abgelesen werden, die auf der nächsthöheren Stufe benötigt wird. Ist das äußerste

Projektorelement erfolgreich geprüft, ist die Spezifikation konsistent und man hat das vollständig konsistente Präsentationsintervall des äußersten Projektors bestimmt.

2. **Konsistenzsicherung:** Bei der Konsistenzsicherung werden die vollständig konsistenten Dienstgütebereiche bestimmt. Dieses Verfahren wird, beginnend mit dem am weitesten außen liegenden Datenraum, entsprechend der Schachtelung absteigend durchgeführt. Für das Dokument in Abbildung 5-5 kann beispielsweise die Reihenfolge T_1, T_2, T_6, T_7, T_8, T_3, T_4, T_5 gewählt werden. Bei der Konsistenzsicherung werden Einschränkungen für die Längen von Intervallen und Verzögerungen aus weiter außen liegenden Datenräumen nach innen weitergegeben. Für jedes Projektorelement wird dabei zuerst die Projektion betrachtet und dann sein Datenraum.

   Für eine Projektion wird der entsprechende Ereignisgraph gebildet, wobei die vollständig konsistente Präsentationsintervallänge gesetzt wird, die auf der nächsthöheren Stufe ermittelt wurde. Durch Anwendung des Längenbestimmungsalgorithmuses berechnet man die vollständig konsistente Länge des Datenraums.

   Ist das Projektorelement ein Multimedia-Projektor, dann werden nun mögliche Einschränkungen für Elemente im Datenraum berechnet. Hierzu wird der Ereignisgraph für die enthaltene Spezifikation gebildet, bestehend aus den Präsentations- und Interaktionsintervallen sowie Intervalloperatoren. In diesen Ereignisgraph werden Knoten und Kanten eingefügt, um die Einschränkungen durch die Länge des umgebenden Datenraums zu repräsentieren. Falls nicht schon enthalten, werden Knoten für den Anfang und das Ende des Datenraums eingefügt und mit einer Kante verbunden, welche die berechnete Länge des Datenraums repräsentieren. Des weiteren wird zwischen dem Anfangsknoten des Datenraums und den Anfangsknoten von Präsentationsintervallen eine Kante der Länge $[0, \infty]$ eingefügt. Zwischen den Endknoten von Präsentationsintervallen und dem Endknoten des Datenraums werden ebenfalls Kanten der Länge $[0, \infty]$ einfügt. Diese Kanten stellen sicher, daß alle Präsentationsintervalle enthaltener Projektorelemente im Datenraum beginnen und enden. Dann wird der Längenbestimmungsalgorithmus angewendet, wodurch man die vollständig konsistenten Intervallängen und Verzögerungen im Datenraum erhält.

Dieser Algorithmus hat den Vorteil, daß der Längenbestimmungsalgorithmus immer nur auf recht kleine Ereignisgraphen angewendet wird. Obwohl der Längenbestimmungsalgorithmus auf jeden zeitlich abgeschlossenen Datenraum zweimal angewendet werden muß, ist dies günstiger als wenn man ihn auf einen großen Ereignisgraph nur einmal anwendet. Ein weiterer Vorteil ist, daß für indeterministische zusammengesetzte Datenräume nicht alle Ereignisgraphen erzeugt und separat getestet werden müssen.

5.3.3 Ermittlung vollständig konsistenter Präsentationsraten

In Tiempo werden Präsentationsraten von Projektionen durch Dienstgütebereiche spezifiziert. Für ein kontinuierliches Medium *o* sind abhängig von der effektiven Präsentationsgeschwindigkeit v^o und seiner natürlichen Präsentationsrate r_0^o nur die effektiven Präsentationsraten r^o mit

$$r^o = \frac{v^o r_0^o}{\Delta u^o}, \qquad \Delta u^o \geq 1, \Delta u^o \in \Re \tag{5-2}$$

möglich. Der effektive Ratenfaktor Δu^o beschreibt, welche Dateneinheiten präsentiert werden. Gilt $\Delta u^o = 1$ werden alle Dateneinheiten präsentiert, für $\Delta u^o = 2$ nur jede zweite Dateneinheit usw. Entsprechend (5-2) kann wie folgt die Korrektheit der spezifizierten Projektionspräsentationsraten evaluiert bzw. der vollständig konsistente Dienstgütebereich ermittelt werden. Der effektive Ratenfaktor Δu^o berechnet sich als Kehrwert des Produkts der spezifizierten Projektionspräsentationsraten der Datenräume, die *o* repräsentieren oder den Datenraum von *o* direkt oder indirekt enthalten. Damit Gleichung (5-2) erfüllt werden kann, muß jede spezifizierte Projektionspräsentationsrate $\tilde{r}^o$ größer null und kleiner oder gleich eins sein.

5.4 Konsistenzprüfung von Reaktionsoperatoren

Die Konsistenz der Reaktionsoperatoren in einem Dokument wird geprüft, nachdem die vollständig konsistenten Dienstgütebereiche für eine Dokumentspezifikation entsprechend dem Verfahren in Abschnitt 5.3 erfolgreich bestimmt worden sind. Reaktionsoperatoren sind konsistent, wenn das Ausführen, der in ihnen enthaltenen Aktionen, zu keinem inkonsistenten zeitlichen Layout einer Präsentation führt und kein Dokumentelement in einen undefinierten Präsentationszustand übergeht.

5.4.1 Präsentationszustände

Der temporale Präsentationszustand *Z* eines Dokuments besteht aus den Präsentationszuständen aller enthaltenen Dokumentelemente. Tabelle 5-1 zeigt, wie die Präsentationszustände von Dokumentelementarten beschrieben werden. In jeder Zustandsbeschreibung gibt der Wert *state* den Elementzustand an. Die möglichen Elementzustände für die Elementarten sind in Abbildung 3-6 zu sehen. Projektoren werden des weiteren durch die aktuelle Präsentationsintervalllänge beschrieben. Bei Intervalloperatoren muß die aktuelle Länge der durch sie implizierten Verzögerungen beachtet werden. Bei Interaktionsmanagern wird die aktuelle Länge des Interaktionsintervalls sowie die Start- und Stoppverzögerung berücksichtigt. In der Zustandsbeschreibung eines Reaktionsoperators beschreibt der Wert *trigger*, ob der Reaktionsoperator aktivierbar ist oder nicht, d.h. die Vor- und Nachbedingungen potentiell erfüllbar sind und enthaltene Aktionen somit ausgeführt werden könnten. Alle anderen Elementarten sind für die Erfassung des temporalen Präsentationszustands eines Dokuments ohne Bedeutung.

Elementart	Zustandsbeschreibung
Projektor	(*state, PresInterval*)
Intervalloperator	(*state, Delay1, Delay2, Delay3*)
Interaktionsmanager	(*state, IntInterval, StartDelay, EndDelay*)
Reaktionsoperator	(*state, trigger*)

Tabelle 5-1: Beschreibung von Präsentationszuständen

Nehmen wir an, der aktuelle Präsentationszustand Z_i enthält q unabhängig aktivierbare Reaktionsoperatoren. Dann könnte in einer Präsentation, je nachdem welcher Reaktionsoperator ausgelöst wird, einer von q neuen Präsentationszuständen Z_{i+k} ($k = 1, ..., q$) entstehen. Jeder dieser Präsentationszustände kann nun wiederum mehrere aktivierbare Reaktionsoperatoren enthalten, wodurch wiederum neue Präsentationszustände entstehen usw. Eine solche Zustandskette bricht ab, falls ein Präsentationszustand keine aktivierbaren Reaktionsoperatoren enthält. Es kann jedoch auch Zustandsketten geben, die nie enden. Da eine Präsentation jedoch nur endlich viele Präsentationszustände einnehmen kann, müssen irgendwann Präsentationszustände auftreten, die äquivalent zu bereits existierenden Zuständen sind.

Def.: Der Präsentationszustand Z_i ist **äquivalent** zu Z_j, wenn für alle in Z_i enthaltenen Präsentationszustände z_i^o von Dokumentelementen *o* ein äquivalenter Präsentationszustand z_j^o in Z_j existiert.

Def.: Der Präsentationszustand z_i^o eines Dokumentelements *o* ist **äquivalent** zu z_j^o, wenn je nach Elementart gilt:

Elementart	Äquivalenzbedingung
Projektor	$(state_i = state_j) \wedge (PresInterval_i \subseteq PresInterval_j)$
Intervalloperator	$(state_i = state_j) \wedge (Delay1_i \subseteq Delay1_j) \wedge$ $(Delay2_i \subseteq Delay2_j) \wedge (Delay3_i \subseteq Delay3_j)$
Interaktions-manager	$(state_i = state_j) \wedge (IntInterval_i \subseteq IntInterval_j) \wedge$ $(StartDelay_i \subseteq StartDelay_j) \wedge (EndDelay_i \subseteq EndDelay_j)$
Reaktionsoperator	$(state_i = state_j) \vee (state_i = \text{inactive})$

Def.: Der Präsentationszustand Z_i ist **teilweise äquivalent** zu Z_j, wenn für alle in Z_i enthaltenen Präsentationszustände z_i^o von Dokumentelementen *o* ein teilweise äquivalenter Präsentationszustand z_j^o in Z_j existiert.

Def.: Der Präsentationszustand z_i^o eines Dokumentelements *o* ist **teilweise äquivalent** zu z_j^o, wenn je nach Elementart gilt:

Elementart	Äquivalenzbedingung
Projektor	$(state_i = state_j) \wedge (PresInterval_i \subseteq PresInterval_j)$
Intervalloperator	$(state_i = state_j) \wedge (Delay1_i \subseteq Delay1_j) \wedge$ $(Delay2_i \subseteq Delay2_j) \wedge (Delay3_i \subseteq Delay3_j)$
Interaktions-manager	$(state_i = state_j) \wedge (IntInterval_i \subseteq IntInterval_j) \wedge$ $(StartDelay_i \subseteq StartDelay_j) \wedge (EndDelay_i \subseteq EndDelay_j)$
Reaktionsoperator	$(state_i = \text{active}) \vee (state_i = \text{inactive})$

Der Unterschied von äquivalenten und teilweise äquivalenten Zuständen ist, daß bei letzteren der neue Zustand mehr aktivierbare Reaktionsoperatoren enthalten kann als der alte Zustand.

5.4.2 Konsistenzverfahren für Präsentationszustandsübergänge

Soll die Konsistenz der Reaktionsoperatoren in einem Dokument sichergestellt werden, muß getestet werden, ob alle durch sie verursachten Präsentationszustände konsistent sind. Hierzu durchläuft man alle Zustandsketten bis sie enden oder ein äquivalenter Präsentationszustand erreicht wird. Findet man einen teilweise äquivalenten Präsentationszustand, brauchen nur noch die Zustandsübergänge geprüft zu werden, welche durch die zusätzlich aktivierbaren Reaktionsoperatoren entstehen, da die anderen Zustandsübergänge bereits geprüft wurden. Ausgehend von einem Präsentationszustand werden wie folgt neue Präsentationszustände bestimmt:

Schritt 1: Bestimmung aktivierbarer Reaktionsoperatoren

Man identifiziert die Menge R von Reaktionsoperatoren, die aktiv sind und deren Auslösebedingung erfüllt sein kann, dazu muß der referenzierte Interaktionsmanager aktiv sein. Für jeden Reaktionsoperator r in R werden nun die folgenden Schritte durchgeführt.

Schritt 2: Überprüfung der Dokumentelementzustände

Für den Reaktionsoperator r wird überprüft, ob das Ausführen der in r enthaltenen Aktionen valide Zustandsübergänge bewirkt, ansonsten ist r inkonsistent.

Schritt 3: Bestimmung der Interaktionsintervallänge

Man bestimmt die maximale Interaktionsintervallänge des durch den Reaktionsoperator r referenzierten Interaktionsmanagers m_r. Hierzu werden zusätzliche Kanten in den Ereignisgraph eingefügt.

Man bestimmt die Menge V_r der aktiven Projektoren und Interaktionsmanager, die durch Stopp-, Geschwindigkeitsverändernde- oder Sprungaktionen in r manipuliert werden sollen. Für alle Dokumentelemente in V_r fügt man zwischen dem Anfangsknoten ihres Präsentations- oder Interaktionsintervalls und dem Anfangsknoten des Interaktionsintervalls von m_r Kanten der Länge $[0, \infty]$ ein. Des weiteren führt man für alle Dokumentelemente in V_r zwischen dem Endknoten des Interaktionsintervalls von m_r und dem Endknoten ihres Präsentations- oder Interaktionsintervalls Kanten der Länge $[0, \infty]$ ein. Durch diese Kanten

wird sichergestellt, daß eine Interaktion nur dann ausgelöst werden kann, wenn die beeinflußten Projektoren bzw. Interaktionsmanager auch tatsächlich Daten ausspielen bzw. sensitiv sind.

Nun wird das in Abschnitt 5.3 beschriebene Konsistenzverfahren angewendet. Tritt eine Inkonsistenz auf, ist der Reaktionsoperator inkonsistent, da es keinen Zeitpunkt gibt an dem alle enthaltenen Aktionen gleichzeitig ausgeführt werden können. Enthält der Reaktionsoperator *stop*-Aktionen, muß das Interaktionsintervall von m_r außerdem die Länge Null haben können. Ist dies nicht der Fall, ist der Reaktionsoperator inkonsistent, da es Zeitpunkte im Interaktionsintervall von m_r gibt, zu dem die *stop*-Aktionen nicht ausgeführt werden können. Tritt keine Inkonsistenz auf, fährt man mit folgendem Schritt fort.

Schritt 4: Simulation von Aktionen

Die Abarbeitung der im Reaktionsoperator *r* enthaltenen Aktionen wird simuliert, wodurch ein neuer Präsentationszustand entsteht. Enthält *r* zusammengesetzte Aktionen, wird jeder Präsentationszustand, der aus einem externen Präsentationskontext resultiert, separat betrachtet.

Für *activate*-Aktionen werden die beeinflußten Dokumentelemente aktiviert, indem entsprechende Knoten und Kanten in den Ereignisgraph eingefügt werden. Für *deactivate*-Aktionen werden die beeinflußten Dokumentelemente deaktiviert, indem die entsprechenden Knoten und Kanten aus dem Ereignisgraph entfernt werden. Bei *start*-Aktionen führt man zwischen dem Endknoten des Interaktionsintervalls von m_r und dem Startknoten des Präsentations- oder Interaktionsintervalls des beeinflußten Projektors oder Interaktionsmanagers eine Kante der Länge Null ein. Bei *stop*-Aktionen führt man zwischen dem Endknoten des Interaktionsintervalls von m_r und dem Endknoten des Präsentations- oder Interaktionsintervalls des beeinflußten Projektors oder Interaktionsmanagers eine Kante der Länge Null ein und setzt die minimale Länge des Präsentations- oder Interaktionsintervalls auf null. Der Effekt einer geschwindigkeitsverändernden Aktion wird simuliert, indem die Länge des Datenraumes des beeinflußten Projektors neu berechnet wird. Die Auswirkung der Aktion wird hierbei für den frühesten und spätesten Zeitpunkt, zu dem sie Auftreten kann, getrennt berechnet:

a. **Frühester Zeitpunkt**

Der früheste Zeitpunkt ergibt sich, wenn der minimale Abstand d_s zwischen dem Datenraumanfangsknoten des beeinflußten Projektors und dem Interaktionsintervallanfangsknoten von m_r aus der aktuellen Längenmatrix gewählt wird. Mit diesem Abstand berechnet man eine neue Länge des Datenraums entsprechend der Formel:

$$\begin{aligned} d_{neu} &= d_s + (d_{alt} - d_s)\frac{v_{alt}}{v_{neu}} \qquad \text{falls} \qquad v_{neu}v_{alt} > 0 \\ d_{neu} &= d_s - d_s\frac{v_{alt}}{v_{neu}} \qquad \text{falls} \qquad v_{neu}v_{alt} < 0 \end{aligned} \tag{5-3}$$

b. **Spätester Zeitpunkt**

Der späteste Zeitpunkt ergibt sich, wenn der maximale Abstand d_s zwischen dem Datenraumanfangsknoten des beeinflußten Projektors und dem Interaktionsintervallendknoten von m_r aus der aktuellen Längenmatrix gewählt wird. Mit diesem Abstand berechnet man nun ebenfalls anhand von Formel (5-3) eine neue Länge des Datenraums.

Durch das Bilden der Vereinigungsmenge der beiden berechneten Datenraumlängen erhält man die aus der geschwindigkeitsverändernden Aktion entstehende Datenraumlänge.

Bei der Simulation eines relativen Sprungs wird ebenfalls die Länge des Datenraums des beeinflußten Projektors neu berechnet. Hierbei wird folgende Formel verwendet:

$$d_{neu} = d_{alt} - \frac{t_{delta}}{v_{alt}} \tag{5-4}$$

Bei der Simulation eines Sprungs zu einem absoluten Zeitpunkt in einem Datenraum, wird die Länge des Datenraums des beeinflußten Projektors entsprechend folgender Formel neu berechnet:

$$d_{neu} = d_s + \left(d_{alt} - \frac{t_{absolut}}{v_{alt}}\right) \tag{5-5}$$

d_s ist hierbei das Längenintervall der Kante zwischen dem Datenraumanfangsknoten des beeinflußten Projektors und dem Interaktionsintervallanfangsknoten von m_r.

Entsprechend dieser Längenänderungen wird nun den Inhalt der Datenräume durch entsprechendes Verlängern oder Verkürzen von Kanten im Ereignisgraph angepaßt.

Sind alle Aktionen simuliert, wird der Längenbestimmungsalgorithmus angewendet. Ergibt sich eine Inkonsistenz, ist der Reaktionsoperator inkonsistent. Enthält der Reaktionsoperator *start*- oder *stop*-Aktionen, muß das Interaktionsintervall von m_r zudem die Länge Null haben können. Ist dies nicht der Fall, ist der Reaktionsoperator inkonsistent, da es Zeitpunkte im Interaktionsintervall gibt, zu dem diese Aktionen nicht ausführbar sind.

Enthält der Reaktionsoperator *activate*-, *deactivate*-, *start*- oder *stop*-Aktionen, wird er deaktiviert, da das nochmalige Ausführen der enthaltenen Aktionen zu invaliden Zustandsübergängen führen würde. Hat ein Interaktionsmanager nur noch deaktivierte Reaktionsoperatoren, wird auch er deaktiviert.

Wurden mittels dieses Algorithmuses alle Präsentationszustände erfolgreich auf Konsistenz geprüft, können die vollständig konsistenten Dienstgütebereiche für zeitliche Attribute bestimmt werden. Hierzu ist für jedes Attribut die Vereinigungsmenge seiner Wertebereiche in den Präsentationszuständen zu bilden.

5.5 Konsistenzverfahren für Dokumente mit Auswahlgruppen

Tiempo-Dokumente können Auswahlgruppen in jedem zusammengesetzten Datenraum enthalten. Abbildung 5-6 zeigt die logische Zuordnung von Auswahlgruppen zu Datenräumen in einem Beispieldokument. Um die Konsistenz einer temporalen Spezifikation mit Auswahlgruppen festzustellen, wird geprüft, ob es für jede enthaltene Präsentationsalternative mindestens eine Aktivierung gibt, bei der das dadurch entstehende temporale Szenario konsistent ist.

Um diese Überprüfung durchführen zu können, benötigt man sogenannte **Alternativenbäume**. Ein Alternativenbaum beschreibt, welche Kombinationen von Präsentationsalternativen in abhängigen Auswahlgruppen möglich sind. Zwei Auswahlgruppen sind abhängig, wenn die eine Auswahlgruppe in einer Präsentationsalternative der anderen Auswahlgruppe enthalten ist. Für das Abhängigkeitskriterium gilt die Transitivität. Präsentationsalternativen in unterschiedlichen Alternativenbäumen können theoretisch in beliebiger Kombination auftreten. Die Wur-

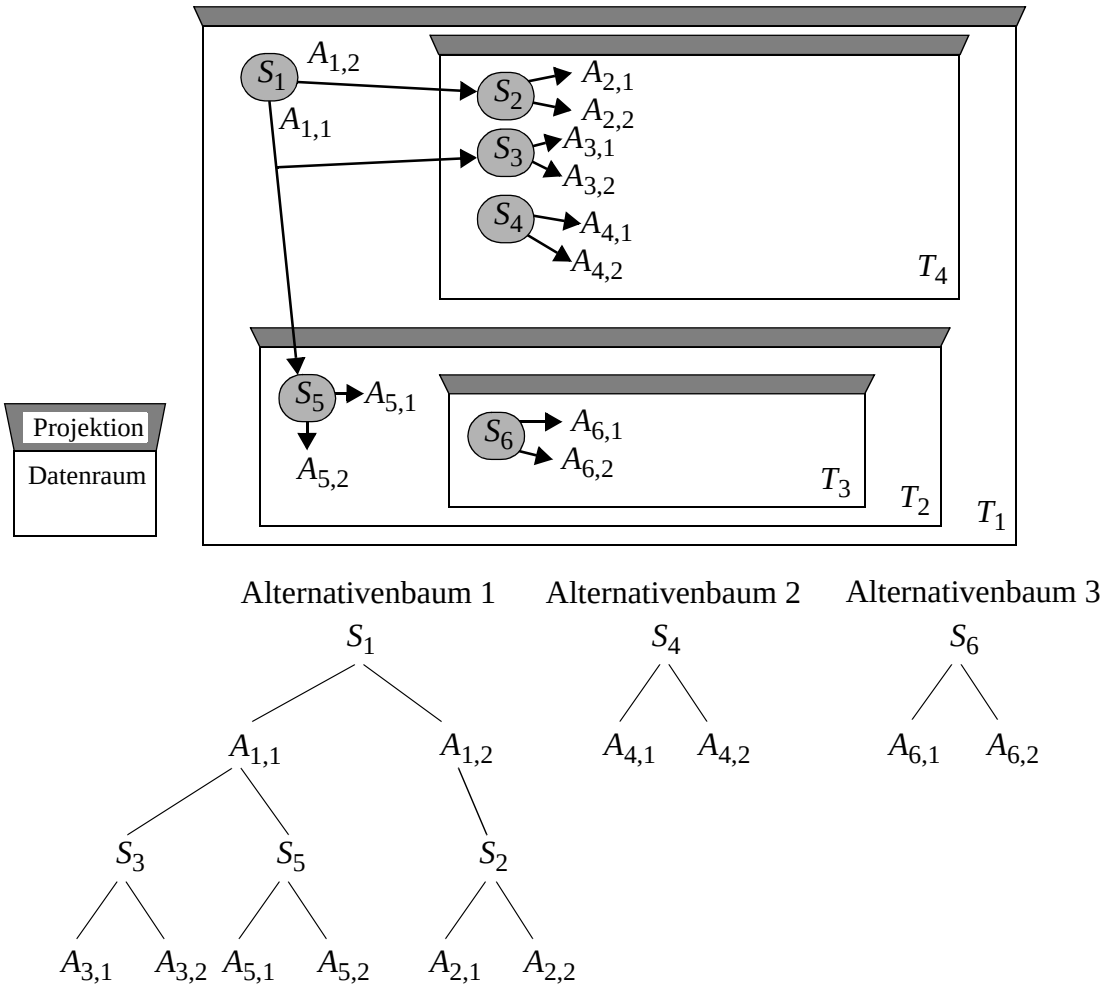


Abb. 5-6 : Konsistenzprüfung von Auswahlgruppen

zel eines Alternativenbaums ist immer eine Auswahlgruppe, die in keiner anderen enthalten ist. Die Kinder eines Auswahlgruppenknotens sind Knoten, welche die Präsentationsalternativen repräsentieren. Falls eine Präsentationsalternative Auswahlgruppen enthält, hat sie entsprechende Auswahlgruppenknoten als Kinder. Unten in Abbildung 5-6 sind die Alternativenbäume zum Dokument in der oberen Hälfte dargestellt. In diesem Beispiel gibt es drei Alternativenbäume mit den Auswahlgruppen S_1, S_4 und S_6 als Wurzel.

Um die möglichen Kombinationen von Präsentationsalternativen in einem Dokument zu berechnen, werden für jeden Alternativenbaum b die Menge K_b der Kombinationen von Präsentationsalternativen bestimmt. Dazu läuft man für jedes Blatt l des Baums b aufwärts bis zur Wurzel und fügt die traversierten Präsentationsalternativen in eine Menge $A_{b,l}$ ein. Die $A_{b,l}$ wer-

den in die Menge K_b eingefügt. Im Beispieldokument in Abbildung 5-6 erhält man die folgenden Mengen K_b:

$$K_1 = \{\{A_{1,1}, A_{3,1}\}, \{A_{1,1}, A_{3,2}\}, \{A_{1,1}, A_{5,1}\}, \{A_{1,1}, A_{5,2}\}, \{A_{1,2}, A_{2,1}\}, \{A_{1,2}, A_{2,1}\}\}$$
$$K_2 = \{\{A_{4,1}\},\{A_{4,2}\}\}$$
$$K_3 = \{\{A_{6,1}\},\{A_{6,2}\}\} .$$

Um alle möglichen Kombinationen K von Präsentationsalternativen im Dokument aufzustellen, bildet man nun alle Kombinationen von Elementen der Mengen K_b, d.h.

$$K = \{\{A_{1,1}, A_{3,1}, A_{4,1}, A_{6,1}\}, \{A_{1,1}, A_{3,2}, A_{4,1}, A_{6,1}\}, \{A_{1,1}, A_{5,1}, A_{4,1}, A_{6,1}\}, ...\} .$$

Zur Konsistenzprüfung werden die in K enthaltenen Präsentationsalternativenmengen M_i der Reihe nach betrachtet. Für jede Menge M_i aktiviert man die in den Präsentationsalternativen enthaltenen Elemente und führt dann, wie in den vorherigen Abschnitten beschrieben, eine Konsistenzprüfung durch. Ist das Präsentationsszenario M_i konsistent, wird dies vermerkt und die berechneten vollständig konsistenten Werteintervalle aller zeitlichen Attribute werden gespeichert.

Nachdem alle Präsentationsalternativenmengen geprüft worden sind, wird überprüft, ob alle im Dokument enthaltenen Präsentationsalternativen in mindestens einer konsistenten Präsentationsalternativenmenge vorkommen. Für Präsentationsalternativen, die in keiner konsistenten Präsentationsalternativenmenge vorkommen, ist die sie enthaltende Auswahlgruppe nicht vollständig konsistent, weil eine ihrer Präsentationsalternativen nicht aktiviert werden kann, ohne Fehler im Dokument zu verursachen. Der vollständig konsistente Wertebereich eines zeitlichen Attributs berechnet sich als Vereinigungsmenge des für jede konsistente Präsentationsalternativenmenge M_i gespeicherten vollständig konsistenten Wertebereichs.

Um bei der Konsistenzprüfung entsprechend Abschnitt 5.3 wiederholtes Berechnen von Datenräumen zu vermeiden, speichert man bei jeder Konsistenzprüfung für T_i die Länge des Präsentationsintervalls l_j in Abhängigkeit der Präsentationsalternativenmenge N_j und erhält somit Tupel (N_j, l_j). Zur Bestimmung von N_j werden aus der aktuellen Präsentationsalternativenmenge M_i alle Alternativen entfernt, die keine in T_i direkt oder indirekt enthaltenen Elemente referenzieren. Bevor eine Konsistenzprüfung für ein T_i durchgeführt wird, wird überprüft, ob

alle Elemente eines N_j in der aktuellen Präsentationsalternativenmenge M_i enthalten sind. Ist dies der Fall, muß die Konsistenzprüfung für T_i und die in T_i enthaltenen Datenräume nicht durchgeführt werden, da l_j die korrekte Länge des Präsentationsintervalls von T_i repräsentiert.

Bei der Konsistenzsicherung entsprechend Abschnitt 5.3 ist eine ähnliche Optimierung möglich. Hier wird für jedes T_i die Länge des Präsentationsintervalls l_j in Abhängigkeit der Menge N_j enthaltener Präsentationsalternativen gespeichert. Die N_j werden genau wie bei der Konsistenzprüfung bestimmt. Die Konsistenzsicherung wird nur durchgeführt, falls man keine Menge N_j findet, deren Elemente in der aktuellen Präsentationsalternativenmenge M_i enthalten sind oder die aktuelle Länge des Präsentationsintervalls größer als l_j ist. Ist die letztere Bedingung erfüllt, wird l_j im Tupel durch die aktuelle Länge des Präsentationsintervalls ersetzt. Auf diese Weise wird immer das längste Präsentationsintervall eines Datenraums, für welches die Konsistenzsicherung bereits durchgeführt wurde, in Abhängigkeit der darin aktiven Präsentationsalternativen gespeichert, wodurch eine Konsistenzsicherung für T_i idealerweise nur einmal für jede enthaltene Kombination von Präsentationsalternativen durchgeführt wird.

5.6 Bewertung

Die vorgestellten Konsistenzverfahren erlauben es, Inkonsistenzen in einer Dokumentspezifikation zu finden. Damit entfällt das manuelle Testen auf Konsistenz, welches bei adaptiven Dokumenten sehr aufwendig ist. Darüber hinaus ermöglichen die Verfahren die Bestimmung der maximalen Dienstgütebereiche von flexiblen Attributen. Eingesetzt in einem Autorensystem sind Autoren somit über die konsistenten Werte flexibler Attribute informiert, wodurch Spezifikationsfehler vermieden werden können.

In existierende Dokumentensysteme wie CHIMP [CPS96], Mbuild [HSR92, HSR94], Firefly [BuZe92, BuZe93] und MODE [BHL91] ist eine Konsistenzprüfung nur durch das Anstoßen des Ablaufplanungsverfahrens möglich und sie integrieren keine speziellen Konsistenzsicherungskonzepte im Bezug auf flexible Spezifikationen. Solange für ein Dokument ein konsistenter Ablaufplan erzeugt werden kann, werden in der Regel jedoch keine Inkonsistenzen angezeigt. Somit kann ein Dokument viele inkonsistente Präsentationsalternativen enthalten, die nie zum Einsatz kommen, den Autor aber glauben lassen, das Dokument ist extrem adaptiv. Durch das vorgestellte Konsistenzsicherungsverfahren wird dies vermieden.

6 Ablaufplanung adaptiver Multimedia Dokumente

Bevor ein Multimedia-Dokument präsentiert werden kann, muß ein **Ablaufplan** erstellt werden. Ein Ablaufplan beschreibt welche Aktionen zu welchen Zeitpunkten vom Präsentationssystem durchzuführen sind, um eine Multimedia-Präsentation entsprechend der Dokument-Spezifikation zu generieren. Den Prozeß der Erzeugung eines Ablaufplans nennt man **Ablaufplanung**. Abhängig von der Dokumentensprache in der das Multimedia-Dokument vorliegt, ist die Erstellung des Ablaufplans und somit die Ablaufplanung unterschiedlich aufwendig. Bei Dokumentensprachen mit implementierungsnahen Konstrukten, wie beispielsweise MHEG [MHEG94] oder ScriptX [Kale93], ist der Ablaufplan im Prinzip direkt im Multimedia-Dokument enthalten. Im Gegensatz dazu ist die Ablaufplanung bei Dokumentensprachen mit abstrakten Konstrukten, wie HyTime [HyTi92] und Tiempo [Wira97] aufwendiger. Anhand eines Ablaufplans wird die Multimedia-Präsentation erzeugt, indem ein **Präsentationssystem** die enthaltenen Aktionen zum angegebenen Zeitpunkt durchführt.

Die Präsentation kontinuierlicher Medien erfordert viele Ressourcen, wodurch es bei der Präsentation von Multimedia-Dokumenten schnell zu Ressourcenengpässen kommen kann. Ressourcenengpässe mindern die Präsentationsqualität, da beispielsweise Synchronisationsanforderungen nicht mehr erfüllt werden können. Ziel der **adaptiven Ablaufplanung** ist es, bei der Erstellung eines Ablaufplans die Ressourcensituation zu berücksichtigen und Ablaufpläne so zu erstellen, daß Ressourcenengpässe vermieden werden.

Bevor in Abschnitt 6.2 Ansätze für adaptive Ablaufplanungsverfahren erörtert werden, werden in Abschnitt 6.1 die Aktionen, welche die Abarbeitung eines Ablaufplans impliziert, und ihr Ressourcenbedarf näher betrachtet.

6.1 Präsentationsaktionen und deren Ressourcenbedarf

Die nun folgende Betrachtung basiert auf der in Abbildung 6-1 dargestellten verteilten Präsentationsumgebung. Multimedia-Dokumente sowie Medien sind auf Dokument-Servern gespeichert. Soll ein Dokument oder ein Medium präsentiert werden, liest eine Zugriffskomponente *C* die entsprechenden Dateneinheiten vom Speichermedium (Festplatte, CDROM, DVD etc.) und

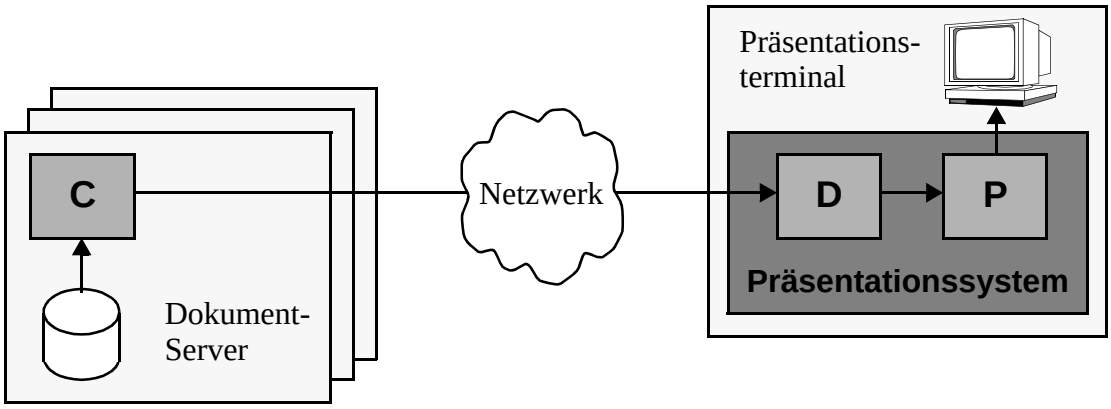


Abb. 6-1 : Verteilte Präsentationsumgebung

schickt sie über das Netzwerk zum Präsentationsterminal. Auf dem Präsentationsterminal werden die Dateneinheiten gepuffert, bis sie benötigt werden. Ist dies der Fall, werden sie von einer Dekodierungskomponente *D* dekomprimiert und an eine Darstellungskomponente *P* weitergereicht, welche die Dateneinheiten ausspielt. Abweichungen von dieser Architektur sind möglich, beispielsweise kann die Dekodierungskomponente fehlen, wenn die Dateneinheiten unkomprimiert übertragen werden. Es kann zusätzlich eine Komprimierungskomponente auf dem Dokument-Server hinzukommen, wenn ein Medium nicht in komprimierter Form gespeichert ist. Aus Gründen der Speichereffizienz werden Medien jedoch meistens komprimiert auf dem Dokument-Server gespeichert sein, und auch in diesem Format übertragen werden.

Der Präsentationsvorgang eines Medienobjekts kann in drei Phasen eingeteilt werden:

1. In der **Vorbereitungsphase** wird die Präsentation des Medienobjekts vorbereitet, um zu einem definierten Zeitpunkt mit der nächsten Phase beginnen zu können.

2. In der **Präsentationsphase** wird das Medienobjekt dem Benutzer präsentiert.

3. Nach Abschluß der Präsentationsphase wird in der **Nachbereitungsphase** die Verarbeitung des Medienobjekts durch Löschen von Verwaltungsinformationen beendet.

Diese drei Phasen müssen nacheinander durchgeführt werden, jedoch nicht unmittelbar aufeinanderfolgen. Beispielsweise kann die Vorbereitungsphase eines Medienobjekts beliebig lange vor dem Beginn seiner Präsentationsphase abgeschlossen sein.

In jeder dieser Phasen sind eine oder mehrere **Präsentationsaktionen** (kurz: Aktionen) durchzuführen. Jede Präsentationsaktion hat einen bestimmten Ressourcenbedarf. Der gesamte Ressourcenbedarf einer Multimedia-Präsentation ergibt sich aus dem Ressourcenbedarf der simultan durchgeführten Präsentationsaktionen.

Hinsichtlich des Zeitpunkts zu dem ein Ressourcenbedarf von der Präsentationsumgebung erfüllt werden muß, kann man zwei Klassen von Ressourcenanforderungen unterscheiden:

- **Positionsabhängige Ressourcenanforderungen** müssen zu einem vorgegebenen Zeitpunkt erfüllt werden, sonst kommt es unmittelbar zu Qualitätsminderungen. Zu dieser Klasse gehört beispielsweise der Bedarf an verfügbarem Speicher, CPU-Zyklen und Bandbreite in der Präsentationsphase kontinuierlicher Medien.

- **Positionsunabhängige Ressourcenanforderungen** sind in gewissen Grenzen verschiebbar. Hier kommt es nicht darauf an, daß zu einem bestimmten Zeitpunkt eine bestimmte Ressourcenmenge vorhanden ist, sondern darauf, daß in einem bestimmten Zeitintervall eine bestimmte Ressourcenmenge zur Verfügung steht. Zu dieser Klasse gehört beispielsweise der Bedarf an verfügbaren CPU-Zyklen und Bandbreite in der Vorbereitungsphase von Medien.

Präsentationsaktionen und deren Ressourcenbedarf werden im folgenden einer detaillierten Analyse unterzogen.

6.1.1 Diskrete Medienobjekte

Abbildung 6-2 zeigt die relative zeitliche Anordnung der Präsentationsaktionen, die für ein diskretes Medienobjekt durchzuführen sind, wenn dessen Präsentationsanfang t_b und Präsentationsende t_e gegeben sind:

1. Als erstes sind die Daten des Medienobjekts zum Präsentationsterminal zu transferieren. Dies wird meist mit herkömmlichen verbindungsorientierten Transportprotokollen erfolgen, wie dem auf TCP/IP [Tane88] basierenden HTTP [FGM+97]. Je nach verwendetem Transportprotokoll können hierbei etwas unterschiedliche Aktionen nötig sein. Im allgemeinen ist jedoch:

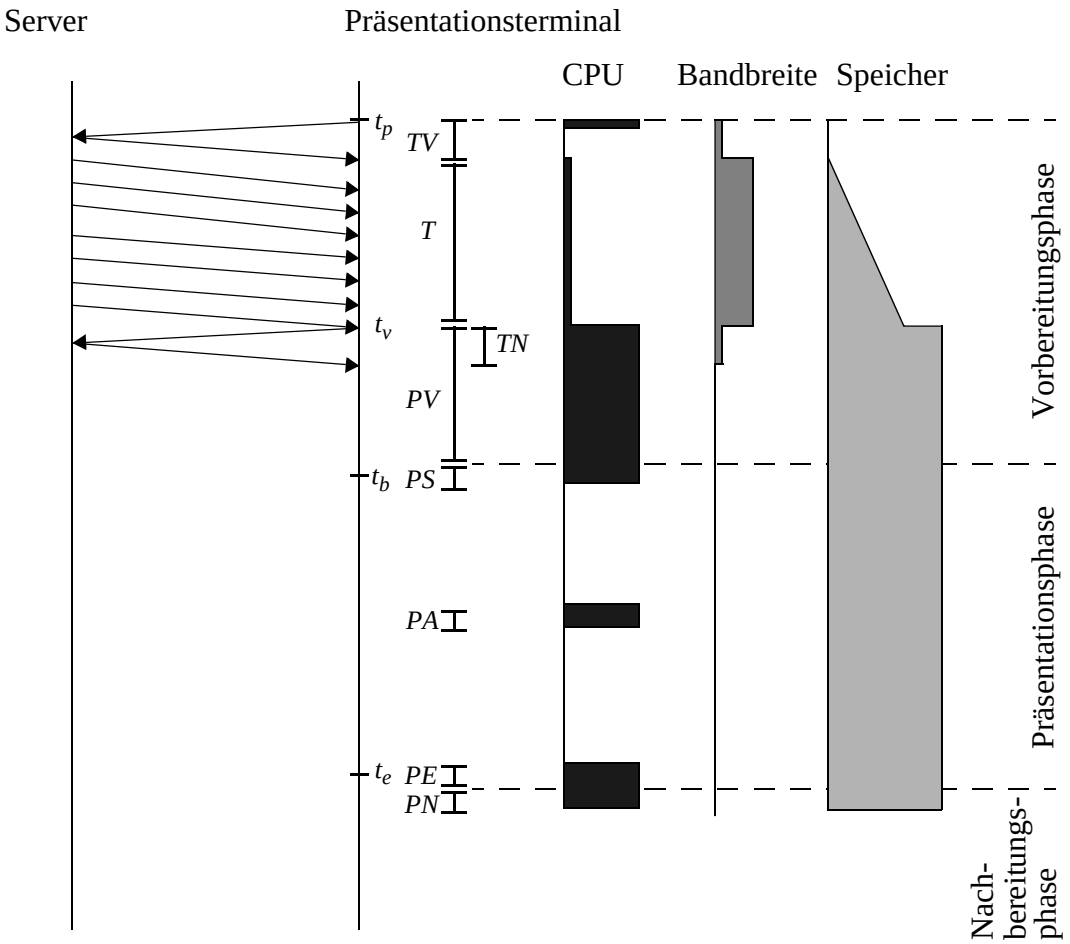


Abb. 6-2 : Präsentationsaktionen bei diskreten Medienobjekten

a. eine Verbindung zum Server aufzubauen und diesem mitzuteilen, welches Medium transferiert werden soll (*TV*).

b. Die vom Server geschickten Daten sind zu empfangen und zu speichern (*T*).

c. Nach erfolgtem Transfer ist die Verbindung abzubauen (*TN*).

Diese drei Phasen werden in der Regel direkt aufeinanderfolgen. Sind die Daten eines diskreten Medienobjekts direkt im Multimedia-Dokument enthalten, entfallen die Aktionen.

2. Die Präsentation der Daten ist vorzubereiten (*PV*). Hierzu gehört einerseits die Dekodierung der transferierten Daten, bei der sie in das Präsentationsformat gebracht werden. Andererseits aber auch Dinge wie die Bestimmung von Masken, so daß die spezifizierten

Überlappungen mit anderen Medienobjekten eingehalten werden. Diese Aktion kann beginnen, sobald alle Mediendaten übertragen sind.

3. Zum Zeitpunkt t_b ist das Ausspielen der Daten anzustoßen (*PS*).

4. Benutzerinteraktionen können eine Manipulation der dargestellten Daten erfordern. Des weiteren kann ein Auffrischen der dargestellten Daten notwendig werden (*PA*).

5. Bei Erreichen von Zeitpunkt t_e ist das Ausspielen der Daten zu beenden (*PE*).

6. Schließlich muß noch die Präsentation des Medienobjekts abgeschlossen werden (*PN*). Hierzu gehört beispielsweise das Freigeben des Speichers mit den Daten des Medienobjekts.

Auf der rechten Seite von Abbildung 6-2 ist schematisch dargestellt, welche Ressourcen die Präsentationsaktionen benötigen. Die Aktion *T* erfordert eine entsprechende Netzwerkbandbreite zwischen Server und Präsentationsterminal zum Transfer der Dateneinheiten. Es sei *V* die Datenmenge des Mediums, welches in *T* zu transferieren ist, $B_a(t)$ die verfügbare Bandbreite zwischen Server und Präsentationsterminal, b_T der Zeitpunkt zu dem *T* beginnt und e_T der Zeitpunkt zu dem *T* endet. Dann muß die folgende Beziehung gelten:

$$V = \int_{b_T}^{e_T} B_a(t)\, dt \qquad (6\text{-}1)$$

Diese Gleichung beschreibt, das *T* solange dauern muß, bis die zum Transfer der Mediendaten zur Verfügung stehende akkumulierte Bandbreite gleich der Datenmenge *V* des Mediums ist.

Bei der Durchführung der Aktion *PV* werden positionsunabhängig CPU-Zyklen benötigt. Unter der Annahme, daß die Aktion *PV* eine fixe Menge an CPU-Zyklen C_{PV} benötigt, muß die folgende Beziehung gelten, um die Aktion ordnungsgemäß durchführen zu können:

$$C_{PV} = \int_{b_{PV}}^{e_{PV}} C_a(t)\, dt \qquad (6\text{-}2)$$

$C_a(t)$ beschreibt die zum Zeitpunkt *t* für die Aktion zur Verfügung stehenden CPU-Zyklen.

Die benötigte Menge Speicher zu jedem Zeitpunkt der drei Phasen ist durch folgende Funktion gegeben:

$$P(t) = \begin{cases} \int_{b_T}^{t} B_a(x)\,dx & \text{falls} \quad b_T \le t < e_T \\ V & \text{falls} \quad e_T \le t < b_{PV} \\ V + V_d & \text{falls} \quad b_{PV} \le t \le e_{PN} \\ 0 & \text{sonst} \end{cases} \tag{6-3}$$

Während der Aktion *T* ist eine ansteigende Menge Speicher zur Pufferung der transferierten Daten erforderlich. Zwischen dem Abschluß der Aktion *T* und dem Beginn der Aktion *PV* ist eine konstante Menge Speicher erforderlich, die der Datenmenge *V* des Medienobjekts entspricht. Bei Beginn der Dekodierung der Mediendaten wird zusätzlich Speicher V_d zur Aufnahme der dekodierten Daten benötigt. Falls sowohl die dekodierten Daten als auch die kodierten Daten während der Präsentation benötigt werden, wird vom Beginn der Aktion *PV* bis Ende der Aktion *PN* eine Speichermenge, wie in Formel (6-3) angegeben, benötigt. Die Speicherung der Daten während der gesamten Präsentationsphase ist manchmal notwendig, um Aktionen *PA* durchführen zu können. Falls nur die dekodierten Daten benötigt werden, wird zwischen dem Ende der Aktion *PV* und dem Ende der Aktion *PN* um *V* weniger Speicher benötigt.

Die Aktion *TV* benötigt Bandbreite und CPU-Zyklen zum Verschicken einer Nachricht. Die Aktion *TN* benötigt Bandbreite und CPU-Zyklen in der gleichen Größenordnung wie *TV*. Die Aktionen *PS*, *PA*, *PE* und *PN* benötigen CPU-Zyklen. Die Ressourcenanforderungen dieser Aktionen sind quantitativ recht schwierig zu erfassen, in der Regel jedoch eher gering.

Im Hinblick auf den Ablaufplan muß für jede Präsentationsaktionen festgelegt sein, wann sie beginnen soll. Um den Aufwand der Ablaufplanung zu reduzieren, werden die folgenden Vereinfachungen vorgenommen: Die Aktion *PS* wird zum Zeitpunkt t_b ausgeführt, dadurch verschiebt sich zwar der tatsächliche Ausspielbeginn um ein paar Millisekunden, was jedoch vernachlässigbar ist. Gleiches gilt für die Aktion *PE*, sie wird zum Zeitpunkt t_e ausgeführt. Die Aktion *PN* wird nach Abschluß von *PE* durchgeführt. Die Aktion *TV* wird zum Zeitpunkt t_p durchgeführt. Die Aktion *PV* wird nach Abschluß der Aktion *T* zum Zeitpunkt t_v durchgeführt. Entsprechend dieser Vereinfachungen muß bei der Ablaufplanung für ein diskretes Medium

neben den durch die Dokumentspezifikation gegebenen Zeitpunkten t_b und t_e noch ein Zeitpunkt t_p und t_v bestimmt werden.

6.1.2 Kontinuierliche Medienobjekte

In Abbildung 6-3 ist die zeitliche Anordnung von Präsentationsaktionen zu sehen, wie sie bei der Präsentation eines kontinuierlichen Medienobjekts auftritt, die zum Zeitpunkt t_b beginnt und zum Zeitpunkt t_e endet. Die Dateneinheiten des kontinuierlichen Medienobjekts werden hierbei mit der Präsentationsrate r ausgespielt. Im einzelnen sind folgende Aktionen durchzuführen:

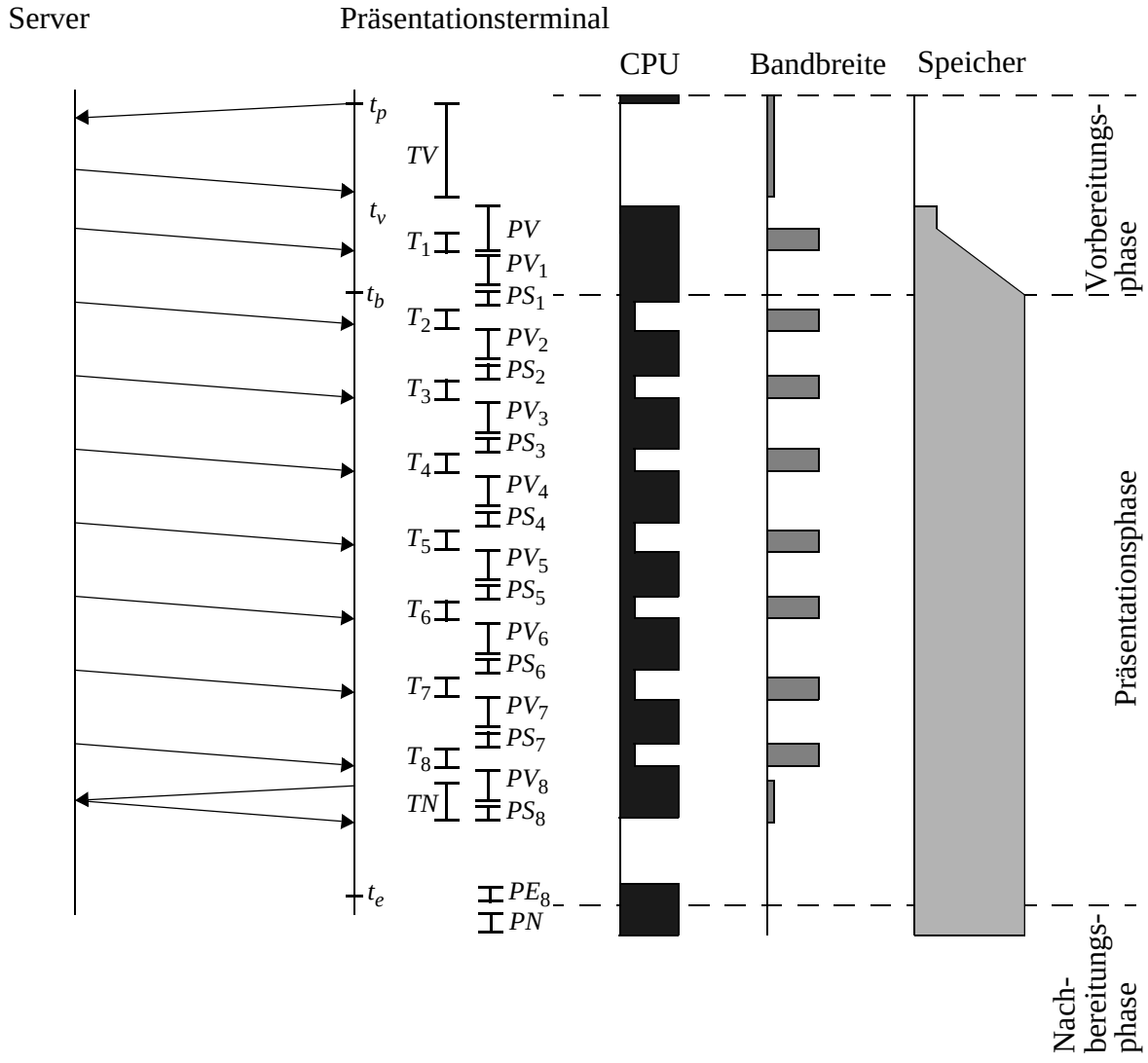


Abb. 6-3 : Präsentationsaktionen bei kontinuierlichen Medienobjekten

1. Als erstes ist dem Server, auf dem die Mediendaten gespeichert sind, mitzuteilen, mit welcher Periode $p = 1/r$ die Dateneinheiten des Mediums zu schicken sind (*TV*).

2. Die Präsentation der Mediendaten muß vorbereitet werden (*PV*), indem beispielsweise Masken für die Ausgabe berechnet werden.

3. Die vom Server transferierten Dateneinheiten i sind zu empfangen (T_i), entsprechend der gegebenen Periode p für die Ausgabe vorzubereiten (PV_i) und auszuspielen (PS_i).

   Die Aktionen T_i können sich mit den Aktionen PV_i und PS_i überlappen. Die Aktionen PV_i und PS_i überlappen sich jedoch in der Regel nicht. Jede Aktion PS_{i+1} beendet die Präsentation von PS_i, d.h. nur die Präsentation der letzten Dateneinheit n wird durch eine explizite Aktion PE_n beendet.

4. Nachdem alle auszuspielenden Dateneinheiten transferiert wurden, kann dem Server das Ende des Transfers mitgeteilt werden (*TN*).

5. Schließlich muß noch die Präsentation des Medienobjekts abgeschlossen werden (*PN*). Hierzu zählt das Freigeben von Speicher.

Auf der rechten Seite von Abbildung 6-3 ist schematisch der durch diese Aktionen verursachte Ressourcenbedarf dargestellt. Die Aktionen *TV* und *TN* benötigen wenig CPU-Zyklen und Bandbreite. Die Aktion *PV* konsumiert CPU-Zyklen und Speicher, der durch die Aktion *PN* wieder freigegeben wird. Die Menge des benötigten Speichers P_{PV} hängt vom Typ des Medienobjekts und dessen Eigenschaften ab, wie beispielsweise der Bildgröße eines Videos. Die Aktionen PE_n und *PN* benötigen wenig CPU-Zyklen.

Aufgrund von unterschiedlichen Verzögerungen in den Übertragungskomponenten zwischen Sender und Empfänger treten beim Transfer von Dateneinheiten über ein Netzwerk in der Regel unterschiedliche Übertragungszeiten auf. Die maximale Varianz der Übertragungszeit d wird als **Übertragungs-Jitter** j bezeichnet. Formal ist Jitter definiert als

$$j = d_{max} - d_{min} \tag{6-4}$$

Um ein periodisches Ausspielen von Dateneinheiten auf dem Präsentationsterminal zu ermöglichen, muß der Jitter durch Puffern von Dateneinheiten ausgeglichen werden. Um Jitter der Größe *j* bei einer Ausspielrate *r* auszugleichen, müssen $2rj$ Dateneinheiten gepuffert werden. Demzufolge muß in der Vorbereitungsphase eines kontinuierlichen Mediums die Datenmenge

$$V = \sum_{i=1}^{2jr} V_i \tag{6-5}$$

transferiert werden. Hierbei ist V_i die Größe der Dateneinheit *i* des Mediums. Da die V_i auf dem Präsentationsterminal in der Regel nicht bekannt sind, ersetzt man diese in Formel (6-5) durch eine Konstante V_u, welche die durchschnittliche Größe einer Dateneinheit des Mediums beschreibt. In der Vorbereitungsphase eines kontinuierlichen Mediums wird somit positionsunabhängig Bandbreite entsprechend folgender Gleichung benötigt:

$$V = \int_{t_p}^{t_b} B_a(t)\, dt \tag{6-6}$$

$B_a(t)$ beschreibt hierbei die verfügbare Bandbreite zwischen Server und Präsentationsterminal zum Zeitpunkt *t.* Im Gegensatz dazu wird in der Präsentationsphase eines kontinuierlichen Mediums positionsabhängige Bandbreite benötigt. Ist b_{T_i} der Zeitpunkt an dem der Transfer der Dateneinheit *i* beginnt und e_{T_i} der Zeitpunkt an dem er endet, dann müssen während der Präsentationsphase für alle Dateneinheiten *i* folgende Beziehungen gelten, damit ein Transfer entsprechend der Periode *p* erfolgen kann:

$$V_i = \int_{b_{T_i}}^{e_{T_i}} B_a(t)\, dt, \quad e_{T_i} - b_{T_i} \le p \tag{6-7}$$

Aufgrund ihrer Komplexität ist diese exakte Formel nicht für die Ablaufplanung geeignet. Man verwendet stattdessen die folgende vereinfachte Formel, die auf der durchschnittlichen Größe V_u einer Dateneinheit des Mediums und der gewählten Präsentationsrate *r* basiert, um den Bandbreitenbedarf eines kontinuierlichen Medienobjekts zu beschreiben:

$$B(t) = \begin{cases} rV_u & \text{falls} \quad b_{T_1} \le t \le e_{T_n} \\ 0 & \text{sonst} \end{cases} \tag{6-8}$$

Bei der Durchführung der Aktionen PV_i und PS_i werden CPU-Zyklen benötigt. Unter der Annahme, daß PV_i bzw. PS_i eine Menge CPU-Zyklen C_{PV_i} bzw. C_{PS_i} benötigen, müssen die folgenden Beziehungen gelten, um die Aktionen zeitgerecht durchführen zu können:

$$C_{PV_i} + C_{PS_i} = \int_{b_{PV_i}}^{e_{PS_i}} C_a(t)\, dt, \quad e_{PS_i} - b_{PV_i} \le p \tag{6-9}$$

$$C_{PS_i} = \int_{b_{PS_i}}^{e_{PS_i}} C_a(t)\, dt, \quad b_{PS_i} = t_b + p(i-1), \quad e_{PS_i} - b_{PS_i} \le \varepsilon \tag{6-10}$$

Hierbei ist $C_a(t)$ die zum Zeitpunkt t für die Aktionen zur Verfügung stehenden CPU-Zyklen. Gleichung (6-9) beschreibt, daß in einer Periode genügend CPU-Zyklen vorhanden sein müssen, um sowohl die Dekodierung als auch das Ausspielen der jeweiligen Dateneinheit durchzuführen. Gleichung (6-10) fordert, daß zum Ausspielzeitpunkt einer Dateneinheit i genügend CPU-Zyklen vorhanden sein müssen, so daß die erlaubte Verzögerung ε beim Ausspielen einer Dateneinheit nicht überschritten wird. Zudem müssen die Aktionen PV_i und PS_i um

$$\Delta t = d_{max} + j \tag{6-11}$$

Zeiteinheiten verzögert zu T_i ausgeführt werden bzw. die Dateneinheiten müssen vom Server um Δt Zeiteinheiten früher abgeschickt werden, um den Jitter ausgleichen zu können.

Da die Formeln (6-9) und (6-10) im Rahmen der Ablaufplanung durch ihre Komplexität kaum handhabbar sind, verwendet man eine vereinfachte Darstellung. Man nimmt an, daß die Aktionen PV_i und PS_i für jede Dateneinheit des Mediums eine annähernd konstante Menge C_u an CPU-Zyklen benötigen. Damit ergibt sich dann der Bedarf an positionsabhängigen CPU-Zyklen während der Präsentationsphase eines Mediums als Produkt der Präsentationsrate r und der Konstante C_u:

$$C(t) = \begin{cases} rC_u & \text{falls} \quad b_{PV_1} \le t \le e_{PS_n} \\ 0 & \text{sonst} \end{cases} \tag{6-12}$$

Bei der Durchführung der Aktion PV werden positionsunabhängig CPU-Zyklen entsprechend Formel (6-2) benötigt.

Hinsichtlich des Speicherbedarfs gibt es vier verschiedene Phasen, die durch folgende Funktion beschrieben werden können:

$$P(t) = \begin{cases} P_{PV} & \text{falls} \quad b_{PV} \le t < b_{T_1} \\ P_{PV} + \int_{b_{T_1}}^{t} B_a(x)\,dx & \text{falls} \quad b_{T_1} \le t < b_{PV_1} \\ P_{PV} + 2jrV_u + V_d & \text{falls} \quad b_{PV_1} \le t \le e_{PN} \\ 0 & \text{sonst} \end{cases} \qquad (6\text{-}13)$$

Hierbei ist P_{PV} der Speicher, der für die Präsentationsvorbereitungen benötigt wird, und V_d die Größe der dekomprimierten Dateneinheiten.

Im Hinblick auf den Ablaufplan muß für jede Präsentationsaktion festgelegt sein, wann sie beginnen soll. Um den Aufwand der Ablaufplanung zu reduzieren, werden die folgenden Vereinfachungen vorgenommen: Die Aktion *TV* wird zum Zeitpunkt t_p begonnen und die Aktion *PV* zum Zeitpunkt t_v. Die Aktionen T_i sind durch den Datentransfer bestimmt. Die Aktion *TN* wird direkt nach T_n durchgeführt. Die Aktion PV_1 wird direkt nach T_{2jr} durchgeführt. Die Aktion PS_1 wird zum Zeitpunkt t_b ausgeführt. Eine Aktion PV_{i+1} wird direkt nach PS_i durchgeführt. Die Aktionen PS_i werden entsprechend der gegebenen Periode bzw. Rate durchgeführt. Die Aktion PE_n wird zum Zeitpunkt t_e ausgeführt. Die Aktion *PN* wird direkt nach Abschluß von PE_n durchgeführt. Entsprechend dieser Vereinfachungen muß bei der Ablaufplanung für ein kontinuierliches Medienobjekt neben den durch die Dokumentspezifikation gegebenen Zeitpunkten t_b und t_e noch ein Zeitpunkt t_p und t_v bestimmt werden.

6.1.3 Zusammengesetzte Medienobjekte

Abbildung 6-4 zeigt die prinzipielle Abfolge von Präsentationsaktionen bei zusammengesetzten Medienobjekten, deren Präsentation zum Zeitpunkt t_b beginnen und zum Zeitpunkt t_e enden soll. Da zusammengesetzte Medienobjekte zwar eine bestimmte Semantik für die enthaltenen Medienobjekte haben, jedoch selbst mit keinen gespeicherten Dateneinheiten assoziiert sind, entfallen bei ihnen Aktionen zum Transfer von Dateneinheiten. Es sind jedoch die folgenden Aktionen durchzuführen:

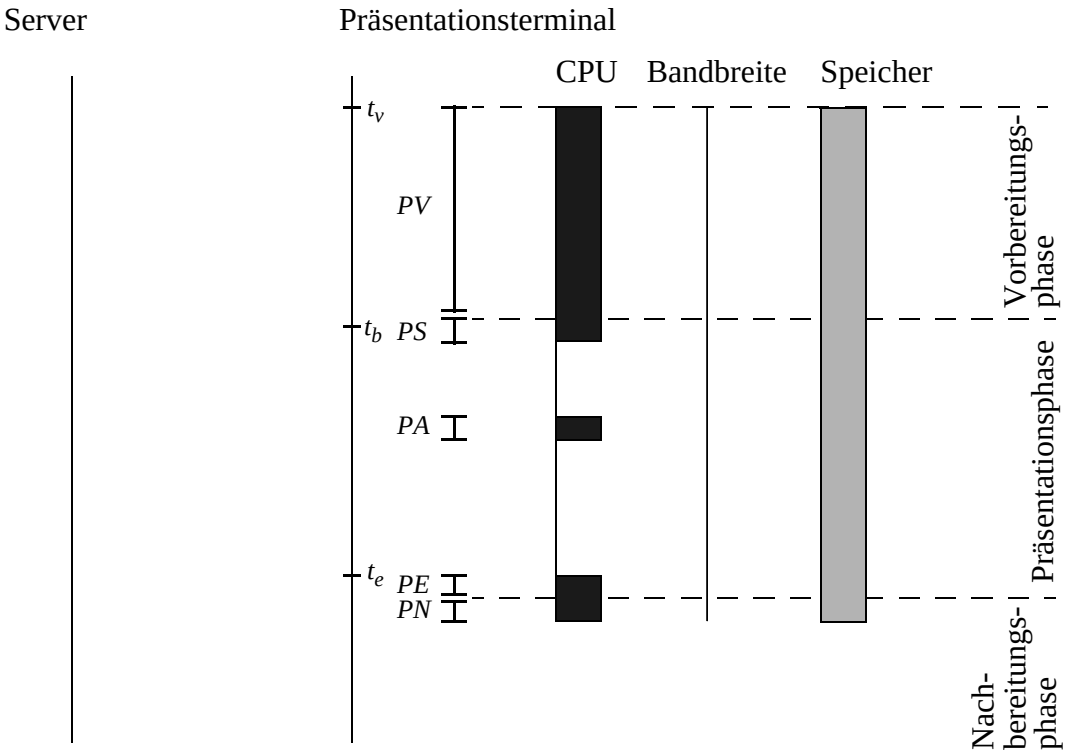


Abb. 6-4 : Präsentationsaktionen bei zusammengesetzten Medienobjekten

1. Als erstes ist die Präsentation des zusammengesetzten Mediums vorzubereiten (*PV*). Hierbei sind beispielsweise bei Fenstern Strukturen der Benutzeroberfläche zu erzeugen.

2. Zum Zeitpunkt t_b ist die Präsentation des Medienobjekts zu initiieren (*PS*).

3. Während das Medienobjekt präsentiert wird, können Benutzerinteraktionen zu verarbeiten sein, welche die Manipulation der Darstellung erfordern (*PA*).

4. Zum Zeitpunkt t_e ist die Präsentation zu beenden (*PE*).

5. Als letztes ist der Präsentationsvorgang zu beenden (*PN*), indem beispielsweise Strukturen der Benutzeroberfläche gelöscht werden.

Auf der rechten Seite von Abbildung 6-4 ist schematisch der Ressourcenverbrauch dieser Aktionen dargestellt. Die Aktion *PV* benötigt eine gewisse Menge CPU-Zyklen C_{PV}. Damit *PV* zeitgerecht durchgeführt werden kann, muß Formel (6-2) eingehalten werden. Während der Aktion *PV* wird eine konstante Menge Speicher P_{PV} allokiert, die durch die Aktion *PN* wieder freigegeben wird. Die Aktionen *PS*, *PA*, *PE* und *PN* benötigen eine geringe Menge an CPU-

Zyklen. Im Rahmen der Ablaufplanung ist für ein zusammengesetztes Medium neben den durch die Dokumentspezifikation gegebenen Zeitpunkten t_b und t_e noch ein Zeitpunkt t_v zu bestimmen.

6.1.4 Dokumente

Abbildung 6-5 zeigt die relative zeitliche Anordnung der Präsentationsaktionen, die für ein Dokument durchzuführen sind:

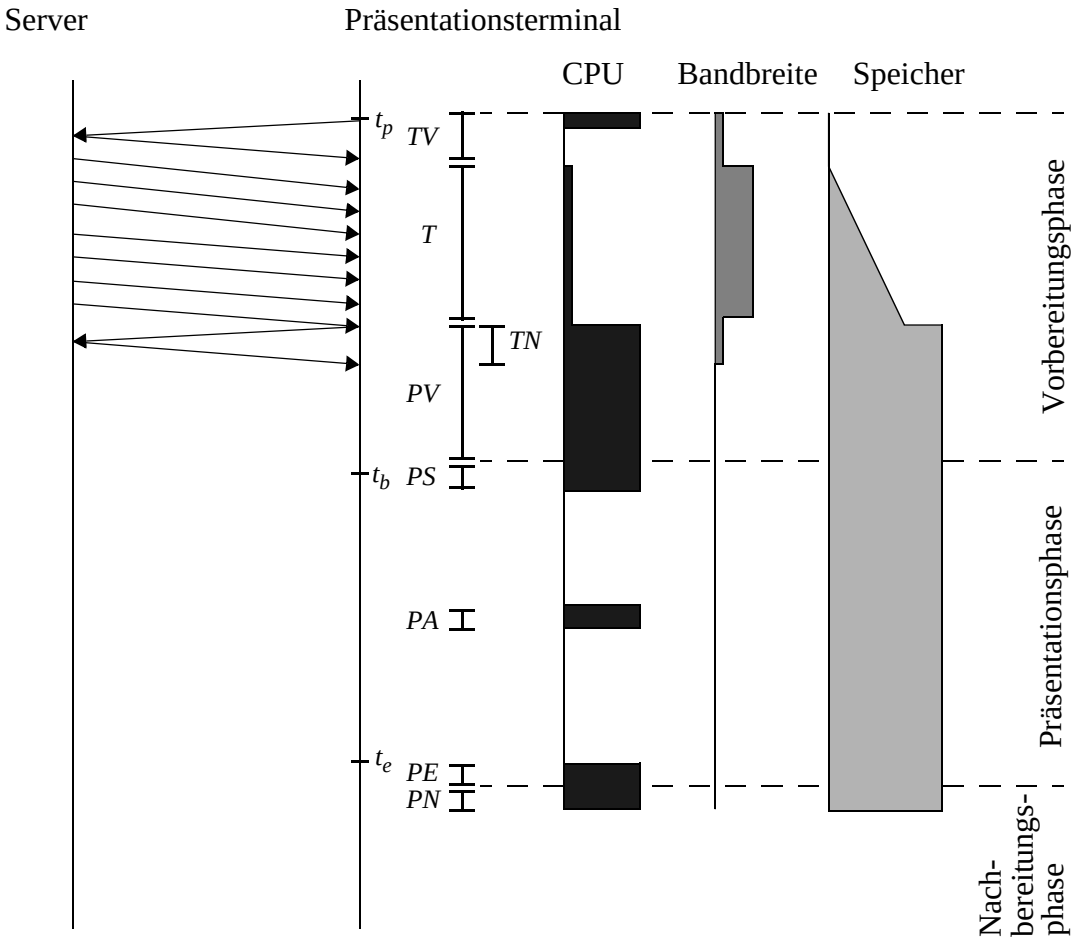


Abb. 6-5 : Präsentationsaktionen bei Dokumenten

1. Als erstes ist das Dokument zum Präsentationsterminal zu transferieren. Dies wird in der Regel mit einem verbindungsorientierten Transportprotokoll, wie HTTP erfolgen. Hierbei ist eine Verbindung zum Server aufzubauen und diesem mitzuteilen, welches Dokument

transferiert werden soll (*TV*). Die vom Server geschickten Daten sind zu empfangen und zu speichern (*T*). Nach erfolgtem Transfer ist die Verbindung abzubauen (*TN*).

2. Die Präsentation des Dokuments ist vorzubereiten (*PV*). Hierzu muß als erstes der für die Präsentation des Dokuments erforderliche Ablaufplan erstellt werden. Dabei wird auch der Startzeitpunkt der Dokumentpräsentation t_b und das potentielle Ende der Präsentation t_e festgelegt.

3. Zum Zeitpunkt t_b ist die Präsentation des Dokuments anzustoßen (*PS*).

4. Benutzerinteraktionen können eine Veränderung des Ablaufplans erfordern (*PA*).

5. Bei Erreichen von Zeitpunkt t_e ist die Dokumentpräsentation zu beenden (*PE*).

6. Schließlich muß die Präsentation durch Löschen des Dokuments aus dem Speicher noch abgeschlossen werden (*PN*).

Auf der rechten Seite von Abbildung 6-5 ist schematisch dargestellt, welche Ressourcen die Präsentationsaktionen benötigen. Wie im Vergleich mit Abbildung 6-2 zu sehen ist, ist der Ressourcenbedarf qualitativ mit dem von diskreten Medienobjekten zu vergleichen. Dementsprechend gelten die Formeln in Abschnitt 6.1.1 auch für Dokumente.

Bei der Durchführung der Aktion *PV* werden CPU-Zyklen C_{PV} und eine Menge Speicher P_{PV} benötigt, in dem u.a. der Ablaufplan abgelegt wird. Die benötigte Menge an Speicher und CPU-Zyklen hängt vom verwendeten Ablaufplanungsalgorithmus und dem Ablaufplanmodell ab und ist allgemeingültig daher nur schwer zu quantifizieren.

Die Aktion *TV* und die Aktion *TN* benötigen CPU-Zyklen und Bandbreite zum Verschicken einer Nachricht. Die Aktionen *PS*, *PA*, *PE* und *PN* benötigen CPU-Zyklen.

6.2 Adaptive Ablaufplanungsverfahren

Die Aufgabe eines adaptiven Ablaufplanungsverfahrens ist es, Präsentationsaktionen so zu planen, daß ausreichend Ressourcen für ihre zeitgerechte Durchführung zur Verfügung stehen.

Formal ausgedrückt bedeutet dies, daß zwischen dem Beginn t_b und Ende t_e der Präsentation für jede benötigte Ressource R gelten muß:

$$\sum_{i=1}^{n} R^i(t) \leq R_a(t), \qquad t_b \leq t \leq t_e \tag{6-14}$$

Ungleichung (6-14) definiert, das zu jedem Zeitpunkt der Präsentation mindestens so viele Ressourcen vorhanden sein müssen, wie die in der Präsentation enthaltenen n Medienobjekte benötigen. Die Funktion $R^i(t)$ beschreibt die Menge der Ressource R, die Medienobjekt i zum Zeitpunkt t benötigt, um entsprechend der Dokumentspezifikation präsentiert werden zu können. Die Funktion $R_a(t)$ beschreibt, wieviele Einheiten der Ressource R zum Zeitpunkt t zur Verfügung stehen. Aufgabe jeder ressourcenorientierten Ablaufplanung ist es, die durch (6-14) gegebene Beziehung bei der Erstellung eines Ablaufplans zu berücksichtigen.

Adaptive Ablaufplanung erfordert nicht zwingend adaptive Dokumente, da Aktionen der Vorbereitungs- und Nachbereitungsphasen je nach Ressourcensituation zeitlich verschoben werden können. Die volle Leistungsfähigkeit wird aber erst bei adaptiven Dokumenten erreicht, da hier auch die Aktionen der Präsentationsphasen angepaßt werden können, wodurch sich erheblich mehr Spielraum für die Einplanung von Aktionen der Vor- und Nachbereitungsphasen ergibt. Adaptive Dokumente schaffen somit durch die enthaltenen Präsentationsalternativen zusätzliche Möglichkeiten die Ungleichungen (6-14) zu erfüllen.

Problematisch an (6-14) ist, daß die Funktionen $R_a(t)$ nur dann im Detail bestimmt werden können, wenn die verfügbaren Ressourcen mittels Reservierung garantiert oder exklusiv benutzbar sind. Sind diese Voraussetzungen nicht gegeben, kann $R_a(t)$ lediglich mit einer gewissen Wahrscheinlichkeit bestimmt werden. Je veränderlicher die Ressourcensituation in der Präsentationsumgebung ist, desto öfter wird sich $R_a(t)$ ändern und demzufolge die $R^i(t)$ angepaßt werden müssen, wodurch sich dann auch der Ablaufplan ändern kann.

Des weiteren stellt sich das Problem, wie man das Einhalten von Ungleichung (6-14) im Rahmen der Ablaufplanung algorithmisch behandeln kann, da (6-14) bei streng mathematischer Formulierung zu einem komplexen Ungleichungssystem führt, dessen Lösung aufwendig und somit zeitintensiv ist.

Adaptive Ablaufplanungsverfahren kann man nach unterschiedlichen Aspekten klassifizieren. Abbildung 6-6 zeigt eine Klassifikation nach den Kriterien wann Adaptionen durchgeführt werden und wann die Vorbereitungsphase durchgeführt wird. Wird nur einmal, bevor die Präsentation beginnt, eine Adaption durchgeführt, sprechen wir von **statischen Ablaufplanungsverfahren**. Werden während der gesamten Präsentation Adaptionen durchgeführt, sprechen wir von **dynamischen Ablaufplanungsverfahren**. Bei diesen Verfahren kann man zwischen **proaktiven** und **reaktive Verfahren** unterscheiden.

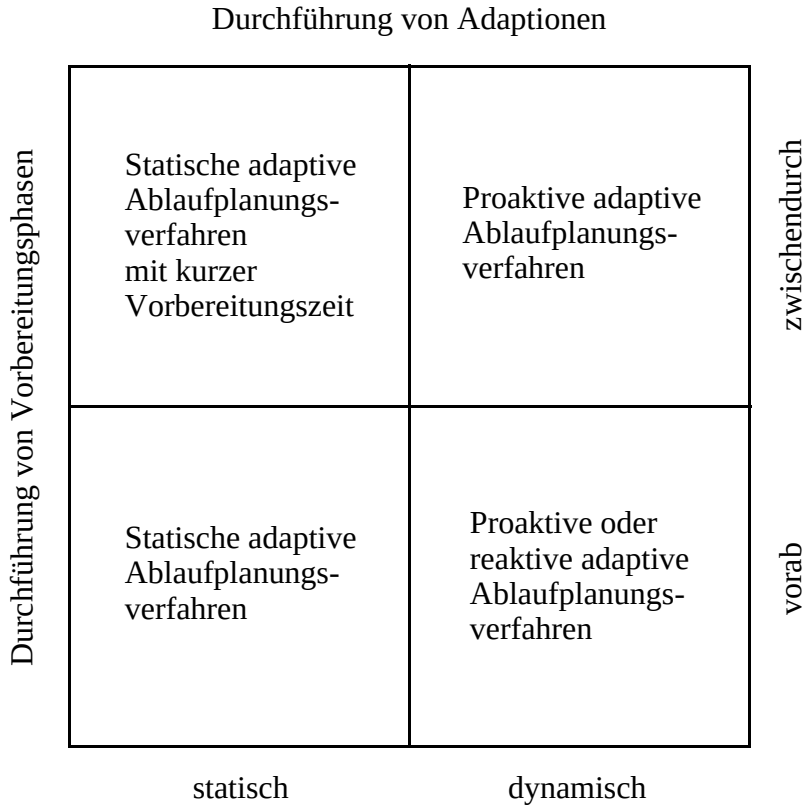


Abb. 6-6 : Klassifikation adaptiver Ablaufplanungsverfahren

6.2.1 Statische Ablaufplanungsverfahren

Bei statischen Ablaufplanungsverfahren wird der Ablaufplan nur einmal unter Berücksichtigung der gegebenen Ressourcensituation berechnet. Statische Ablaufplanungsverfahren eignen sich für Präsentationen bei denen der Ressourcenbedarf der Präsentation sowie das Ressourcenangebot der Präsentationsumgebung bekannt sind und sich während der Präsentation nicht verändern. Dies ist dann gegeben, wenn Ressourcen, beispielsweise durch Reservierung oder eine

exklusiv verwendbare Präsentationsumgebung, garantiert werden können, und keine Interaktionen möglich sind oder der Ressourcenbedarf aller Interaktionen vorab bestimmt werden kann. Ein Anwendungsszenario für statische Ablaufplanungsverfahren ist beispielsweise eine Multimedia-Produktpräsentation als alleinige Anwendung auf einem autarken Computer.

In dieser Klasse folgen alle Ablaufplanungsverfahren demselben Grundprinzip. Die Aktionen der Präsentationsphase von Medienobjekten werden so auf einer Zeitachse plaziert, daß die Dokumentspezifikation eingehalten wird, zu keinem Zeitpunkt mehr Ressourcen benötigt werden, als zur Verfügung stehen und entsprechend der spezifizierten Qualitätsmaße ein optimaler Ablaufplan erreicht ist. Alle Vorbereitungsphasen werden vor Beginn der Präsentation durchgeführt und abgeschlossen. Zu dieser Klasse von Ablaufplanungsverfahren gehört der Ablaufplanungsalgorithmus von MODE [BHL91].

Abbildung 6-7 zeigt den Ablauf einer Dokumentpräsentation bei dem ein statisches Ablaufplanungsverfahren eingesetzt wird. Das Dokument *D* besteht aus einem diskreten Medium *P* und einem kontinuierlichen Medium *V*. Wie in der Abbildung zu sehen ist, wird nach dem Transfer des Dokuments und dessen Dekodierung mit den Vorbereitungsphasen aller enthaltenen Medienobjekte begonnen. Nach deren Abschluß beginnt zum Zeitpunkt t_b die Präsentation des Dokuments und wird zum Zeitpunkt t_e beendet. Da der Beginn und das Ende der Vorbereitungsphasen nicht explizit geplant werden, kann der Start der Präsentationsphase t_b des Dokuments nicht vorab bestimmt werden. D.h. ein entsprechender Ablaufplan wird relative Zeitangaben bezüglich eines bis dato unbekannten Startzeitpunkts beinhalten. Wie Abbildung 6-7 zeigt, wird die Dauer der Vorbereitungsphase durch die limitierte Bandbreite und Menge CPU-Zyklen bestimmt. Während der Präsentationsphase ist der Speicher die limitierende Ressource.

Um den Ablaufplan bei statisch adaptiven Ablaufplanungsverfahren zu bestimmen, muß das Ressourcenangebot und der Ressourcenbedarf der Medienobjekte während der Präsentationsphasen bekannt sein. Bei statisch adaptiven Ablaufplanungsverfahren werden somit nur wenige Informationen über den Ressourcenbedarf von Medienobjekten benötigt. Die Anwendungsszenarien sind jedoch aufgrund der strikten Vorgaben begrenzt und der Speicherbedarf vor Präsentationsbeginn ist sehr hoch, da die Daten aller Medienobjekte vorab auf das Präsentationsterminal transferiert werden müssen. Zudem ist die Verzögerung, mit der die Präsentation letztendlich beginnt, erheblich und kann quantitativ nur schwer bestimmt werden.

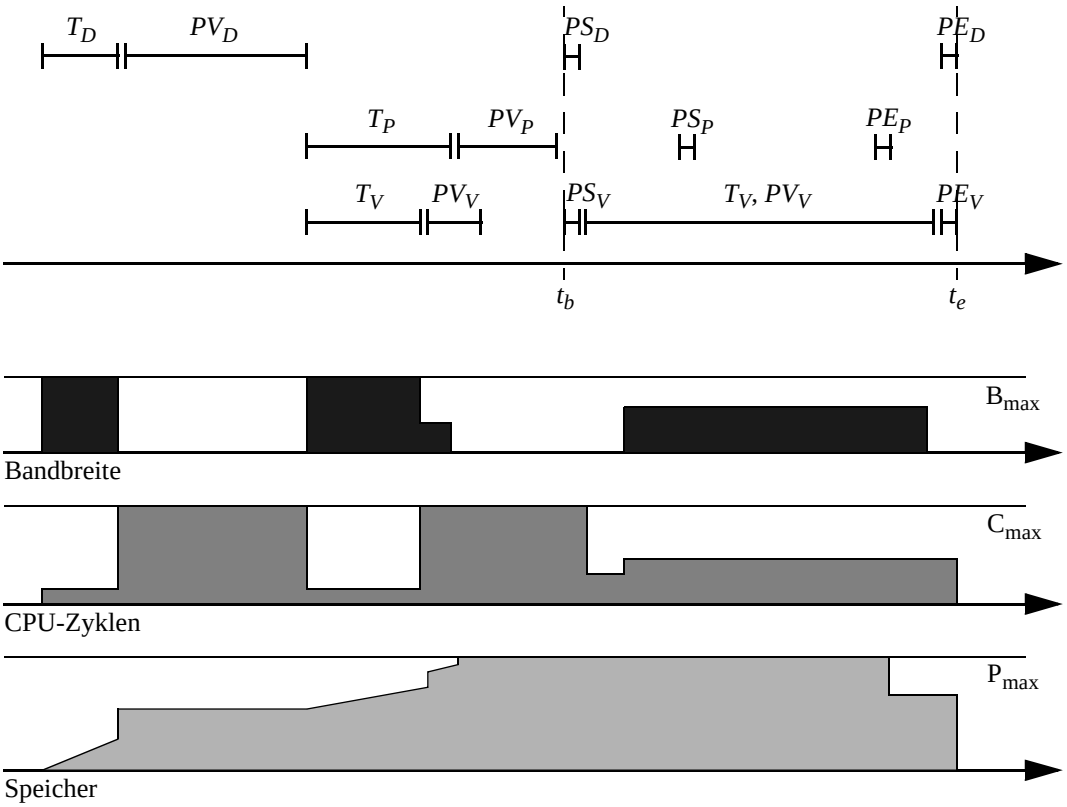


Abb. 6-7 : Statisch adaptive Ablaufplanung

6.2.2 Statische Ablaufplanungsverfahren mit kurzer Vorbereitungszeit

Diese Klasse von Ablaufplanungsverfahren hat die gleichen Randbedingungen wie die vorherige Klasse, auch hier muß der Ressourcenbedarf der Präsentation und das Ressourcenangebot bekannt sein und sich während der Präsentation nicht verändern. D.h. auch hier dürfen Präsentationen keine oder nur eingeschränkte Benutzerinteraktionen enthalten und Ressourcen müssen garantiert werden können.

Bei diesen Verfahren werden Medienobjekte so auf einer Zeitachse plaziert, daß die Dokumentspezifikation eingehalten wird und zu keinem Zeitpunkt mehr Ressourcen benötigt werden als zur Verfügung stehen. Im Unterschied zu der vorherigen Klasse werden jedoch nicht verwendete Ressourcen während der Präsentation zur Durchführung von Vorbereitungsphasen später präsentierter Medienobjekte verwendet. D.h. die Präsentationsaktionen der Vorbereitungsphasen von Medienobjekten werden explizit eingeplant.

Abbildung 6-8 zeigt den Ablauf der Präsentation desselben Dokuments wie in Abschnitt 6.2.1 unter Einsatz eines Ablaufplanungsverfahrens dieser Klasse. Wie der Vergleich mit Abbildung 6-7 zeigt, wird nach dem Transfer des Dokuments und dessen Dekodierung mit der Vorbereitungsphase des kontinuierlichen Mediums begonnen. Die Vorbereitungsphase des diskreten Mediums hingegen überlappt mit der Präsentationsphase des Dokuments.

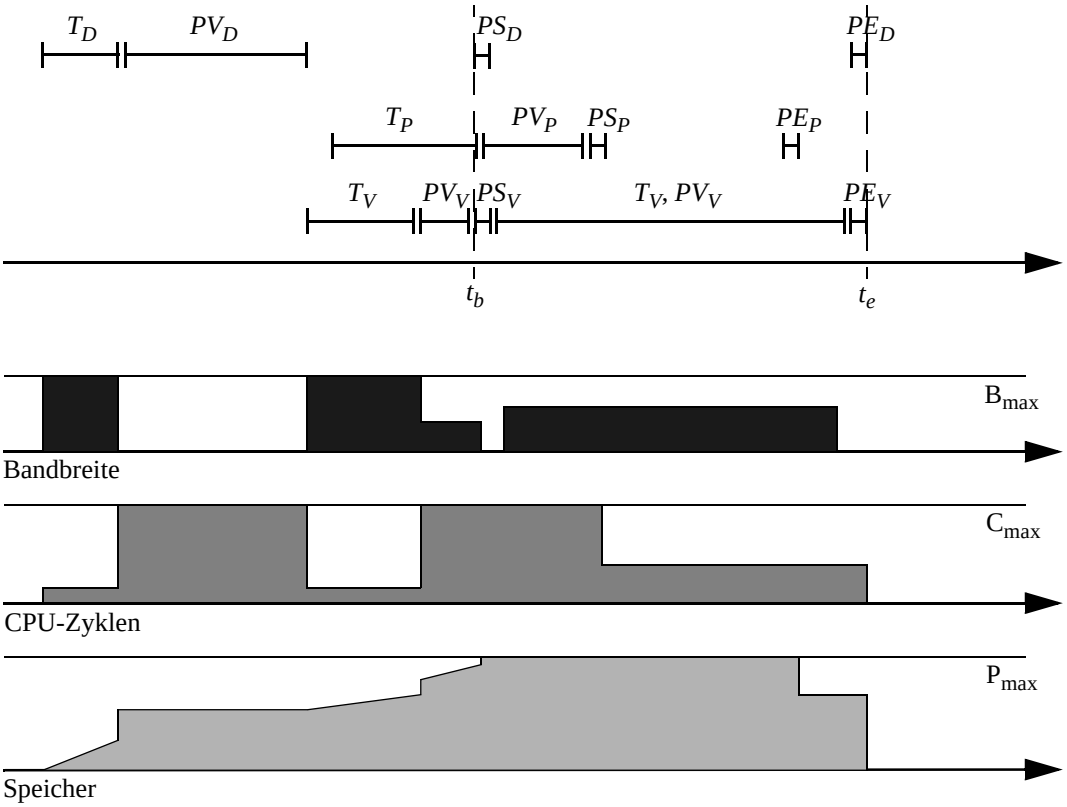


Abb. 6-8 : Statisch adaptive Ablaufplanung mit kurzer Vorbereitungszeit

Wesentliche Vorteile dieser Verfahren sind die verkürzte Zeit zwischen Initiierung der Präsentation und deren tatsächlichem Beginn und der reduzierte Speicherbedarf vor Beginn der Präsentation. Ein weiterer Vorteil ist, daß aufgrund der expliziten Planung der Vorbereitungsphasen, der Startzeitpunkt der Präsentation t_b vorab bestimmbar ist. Der Nachteil dieser Verfahren ist jedoch, daß man zur expliziten Einplanung der Aktionen in Vorbereitungsphasen deren Ressourcenbedarf kennen muß.

Der Ablaufplanungsalgorithmus von CHIMP [CPS96] gehört zu dieser Klasse von Ablaufplanungsverfahren, da er keine dynamische Veränderung eines einmal erstellten Ablaufplans vorsieht. Das Verfahren selbst würde dies jedoch prinzipiell erlauben.

6.2.3 Reaktive Ablaufplanungsverfahren

Diese Klasse von dynamischen Ablaufplanungsverfahren ist anwendbar, wenn sich sowohl der Ressourcenbedarf als auch das Ressourcenangebot während der Präsentation verändern kann. D.h. es können interaktive Dokumente in einem Umfeld mit nicht garantierten Ressourcen präsentiert werden.

Reaktive Ablaufplanungsverfahren sind dadurch charakterisiert, daß sie ohne Ressourceninformationen auskommen. Bei diesen Verfahren wird das Ausspielen von Medienobjekten überwacht, um Abweichungen in der zeitlichen Synchronisation erkennen zu können, wie beispielsweise das verspätete Ausspielen einzelner Dateneinheiten von Videos oder Audios. Beim Auftreten derartiger Synchronisationsfehler wird versucht, das dafür verantwortliche Ressourcenproblem zu beheben, indem beispielsweise die Präsentationsrate eines kontinuierlichen Medienobjekts um einen bestimmten Wert gesenkt wird. Erfolgt das Ausspielen der Präsentation wie geplant, wird davon ausgegangen, daß mehr Ressourcen zur Verfügung stehen, als zur Zeit durch die Präsentation benötigt werden. Wird die Präsentation im Hinblick auf Qualitätsmaße zur Zeit nicht optimal durchgeführt, nähert man sich durch die Wahl anderer Präsentationsalternativen an die maximale Menge verfügbarer Ressourcen an. Ein derartiges Verfahren wird beispielsweise in [ThKl96] vorgeschlagen.

Diese Art von Adaption bringt jedoch einige Restriktionen mit sich. Aufgrund der fehlenden Ressourceninformationen können Adaptionen quantitativ nur sehr schwer bestimmt werden. Wird beispielsweise ein Video ausgespielt und einzelne Dateneinheiten werden zu spät ausgespielt, stellt sich die Frage, um wieviel man die Präsentationsrate senken muß, um den Ressourcenengpaß zu beheben. Senkt man sie zu wenig, wird man weitere Reduzierungen durchführen müssen. Senkt man sie zu viel, wird man sie in kurzer Zeit wieder erhöhen müssen. Dadurch kann leicht ein Schwingen entstehen, das der Präsentationsqualität abträglich ist und dem man deshalb durch entsprechende Mechanismen entgegenwirken muß.

Ein weiteres Problem besteht hinsichtlich der Bestimmung von Zeitpunkten bei denen das Ausspielen eines Medienobjekts beginnen soll. Da man keine Ressourceninformationen verwendet, weiß man nicht, ob man beispielsweise ein optionales Medienobjekt präsentieren kann oder nicht. Fängt man mit dem Ausspielen an und es entsteht dadurch ein Ressourcenengpaß, muß man das Ausspielen wieder beenden. Reaktive Ablaufplanung ist daher nur sinnvoll einsetzbar, wenn Dokumente keine Flexibilität auf Objektebene beinhalten und Startzeiten von Medienobjekten nicht flexibel sind.

Außerdem erlaubt das Fehlen von Ressourceninformationen es nicht, daß Vorbereitungsphasen kontrolliert während der Präsentation durchgeführt werden, da man nicht weiß, wann man damit beginnen muß, um rechtzeitig fertig zu werden. D.h. die Vorbereitungsphasen von Medienobjekten müssen vor Beginn der Präsentationsphase des Dokuments durchgeführt werden.

Zusammenfassend kann man feststellen, daß reaktive Ablaufplanungsverfahren nur sinnvoll zur Adaption von Präsentationsraten während der Präsentationsphase kontinuierlicher Medienobjekte eingesetzt werden können. Sollen andere Möglichkeiten der Adaption berücksichtigt werden, sind zusätzliche Mechanismen erforderlich.

Abbildung 6-9 zeigt die Präsentation eines Dokuments bei dem während der Präsentationsphase kontinuierlicher Medienobjekte ein reaktives Ablaufplanungsverfahren eingesetzt wird. Wie zu sehen ist, führt der Einbruch der Bandbreite während der Präsentationsphase des kontinuierlichen Medienobjekts dazu, daß die Präsentationsgeschwindigkeit zu stark gesenkt wird und erst nach einer recht langen Zeit wieder auf den optimalen Wert zurückgesetzt wird.

6.2.4 Proaktive Ablaufplanungsverfahren

Im Gegensatz zu reaktiven Verfahren wird bei proaktiven Verfahren eine Adaption auf Basis von Ressourceninformationen durchgeführt. Bei diesen dynamischen Verfahren wird ständig zwischen Ressourcenangebot und -bedarf der Präsentation verglichen. Tritt ein Mißverhältnis auf, werden geeignete Adaptionen bestimmt.

Abbildung 6-10 zeigt den Ablauf einer Dokumentpräsentation mit proaktiver Ablaufplanung. Auch hier tritt während der Präsentationsphase des kontinuierlichen Mediums eine Ressourcenknappheit bei der Bandbreite ein. Vergleicht man Abbildung 6-10 mit Abbildung 6-9 stellt man

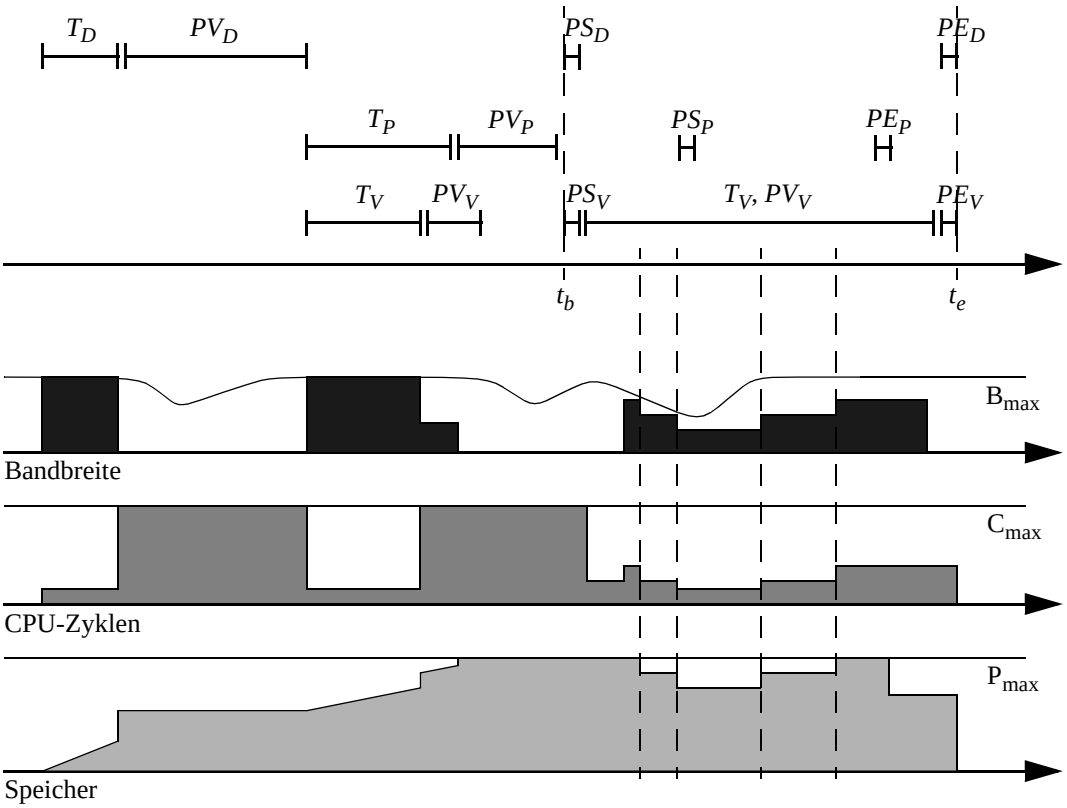


Abb. 6-9 : Reaktive Ablaufplanung

aber fest, daß der Bandbreitenbedarf nur soweit wie nötig reduziert wird. Mit zunehmender Verfügbarkeit von Bandbreite erhöht sich auch der Bandbreitenbedarf der Präsentation entsprechend. Diese enge und rasche Anpassung an Ressourcenänderungen ist aufgrund der Verwendung von Ressourceninformationen möglich, wodurch zu jedem Zeitpunkt die bestmögliche Ressourcenausnutzung gegeben ist.

Diese Klasse von Planungsverfahren erlaubt es, Vorbereitungsphasen während der Präsentation durchzuführen. Wie in Abbildung 6-10 zu sehen ist, finden Vorbereitungsaktionen des diskreten Mediums während der Präsentationsphase des Dokuments statt. Um Vorbereitungsphasen geeignet positionieren zu können, ist es nicht ausreichend das Ressourcenangebot zum momentanen Zeitpunkt zu kennen. Das Ressourcenangebot muß über den gesamten Präsentationszeitraum bekannt sein. In einer Präsentationsumgebung ohne garantierte Ressourcen müssen daher Prognosen erstellt werden.

Proaktive Ablaufplanungsverfahren haben neben den beschriebenen Vorteilen das Problem, daß aufgrund des dynamischen Präsentationsumfelds das Ressourcenangebot in der Zukunft nur

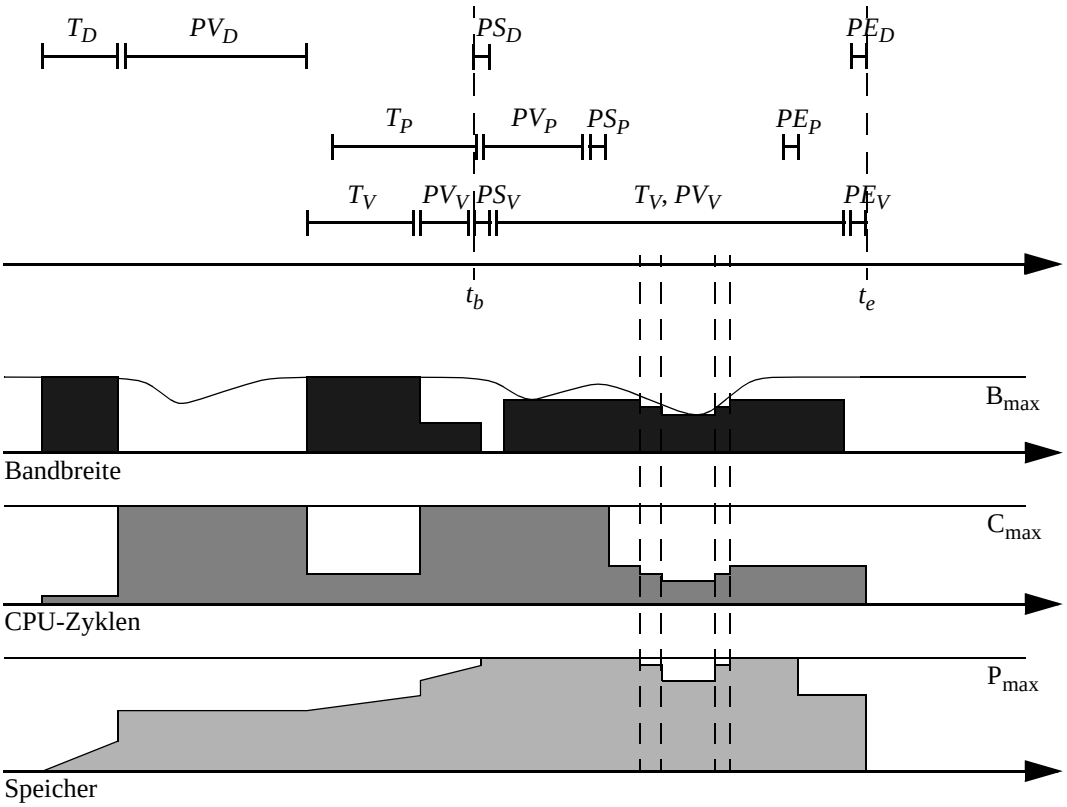


Abb. 6-10 : Proaktive Ablaufplanung

geschätzt werden kann, wodurch Abweichungen von der tatsächlich vorhandenen Ressourcenmenge auftreten können. Diese Abweichungen müssen wiederum durch Adaptionen ausgeglichen werden.

6.3 Zusammenfassung

Die in diesem Abschnitt vorgestellten Ansätze für Ablaufplanungsverfahren haben alle Vor- und Nachteile. Es scheint jedoch, als ob der proaktive Ansatz am flexibelsten hinsichtlich der Einsatzgebiete und des Präsentationsumfelds ist. Das in den nächsten Kapiteln vorgestellte Tiempo-Ablaufplanungsverfahren folgt daher dem proaktiven Ansatz.

7 Gewinnung von Ressourceninformationen

Das Tiempo-Ablaufplanungsverfahren gehört zur Klasse der proaktiven adaptiven Ablaufplanungsverfahren. Dementsprechend wird der Ablaufplan auf der Basis von Informationen über die Ressourcenanforderungen der Multimedia-Präsentation und von Informationen über die verfügbaren Ressourcen der Präsentationsumgebung erstellt. Das Tiempo-Ablaufplanungsverfahren berücksichtigt hierbei die Ressourcentypen Bandbreite, CPU-Rechenzeit und Speicher.

In Abschnitt 7.1 wird die Gewinnung der benötigten Ressourceninformation über die Multimedia-Präsentation erörtert und in Abschnitt 7.2 die Vorhersage verfügbarer Ressourcen der Präsentationsumgebung betrachtet.

7.1 Bestimmung der Ressourcenanforderungen von Medienobjekten

Die Grundlage zur Bestimmung der Ressourcenanforderungen eines Medienobjekts sind seine Eigenschaften. Man kann zwei Klassen von Ressourcenanforderungen unterscheiden:

- **Systemunabhängige Ressourcenanforderungen** können ausschließlich anhand der Eigenschaften eines Mediums bestimmt werden. Zu dieser Klasse gehören die Anforderungen hinsichtlich Bandbreite zum Transfer und hinsichtlich Speicher zur Pufferung von kodierten Mediendaten. Systemunabhängige Ressourcenanforderungen können bereits bei der Erstellung eines Multimedia-Dokuments erfaßt und im Dokument abgelegt werden.

- **Systemabhängige Ressourcenanforderungen** sind durch die Eigenschaften eines Mediums, der Implementierung von Dekodierungs- und Darstellungskomponenten sowie des Präsentationsterminals bestimmt. Zu dieser Klasse gehören die Anforderungen hinsichtlich der benötigten CPU-Zyklen zur Dekodierung und Präsentation von Mediendaten sowie zur Speicherung dekodierter Mediendaten. Systemabhängige Ressourcenanforderungen müssen teilweise durch Leistungsmessungen bestimmt werden, beispielsweise mit Hilfe eines im Präsentationssystem integrierten Testbeds.

In Abhängigkeit der Klasse eines Medienobjekts benötigt das Tiempo-Ablaufplanungsverfahren unterschiedliche Informationen.

7.1.1 Diskrete Medien

Für ein diskretes Medium *o* müssen die Ressourcenanforderungen in Tabelle 7-1 bekannt sein.

Konstante	Semantik
V^o	Benötigtes Transfervolumen des Medienobjekts
C^o	Benötigte CPU-Zyklen zur Dekodierung der Mediendaten
P^o	Benötigte Speichermenge zur Dekodierung der Mediendaten

Tabelle 7-1: Ressourcenanforderungen diskreter Medien

Das benötigte Transfervolumen V^o ist gleich dem Datenvolumen des Mediums. V^o ist eine systemunabhängige Ressourcenanforderung, die bereits während des Spezifikationsprozesses bekannt ist. V^o wird idealerweise im Multimedia-Dokument in einem Attribut des Medienelements spezifiziert. Der in Abschnitt 4 vorgestellte Tiempo-Editor sorgt für eine automatische Erfassung und Spezifikation systemunabhängiger Attributwerte bei der Dokumenterstellung.

P^o kann, je nach Implementierung des Dekodierungsverfahrens und der Darstellungskomponente für einen Medientyp, variieren, und ist damit eine systemabhängige Ressourcenanforderung. Beispielsweise kann die Speichermenge für ein dekodiertes Bild aus der Bildgröße und der Anzahl der Farben pro Bildpunkt als $P^o = Höhe \cdot Breite \cdot Farbtiefe$ bestimmt werden. Hierbei ist die *Farbtiefe* von der Implementierung der Darstellungskomponente abhängig.

C^o ist eine systemabhängige Ressourcenanforderung, die in Abhängigkeit der Implementierung des Dekodierungsalgorithmuses und der Leistungscharakteristik des Präsentationsterminals bestimmt werden muß. Da man für eine proaktive Ablaufplanung den Ressourcenbedarf anhand der Dokumentspezifikation ermitteln können muß, ist es erforderlich für jeden Medientyp eine Formel zu finden, mittels der man C^o aus den Medieneigenschaften berechnen kann. Dazu muß man in der Regel Messungen durchführen.

In Tabelle 7-2 sind die Ergebnisse solcher Messungen für die Darstellungskomponente von Bildern im Tiempo-Präsentationssystem zu sehen. Gemessen wurde hierbei die Anzahl der CPU-Zyklen (Spalte 6), die für die Dekodierung von Bildern benötigt wird. Betrachtet man die Eigen-

schaften der Bilder hinsichtlich Kodierung, Bildgröße und Datenvolumen, stellt man fest, daß das Datenvolumen keinen Einfluß hat, wohl aber die beiden anderen Faktoren. Dividiert man die gemessenen CPU-Zyklen durch die Breite und Höhe eines Bildes, erhält man in Abhängigkeit der Kodierung *t* immer eine ähnliche Dekodierungskennzahl C_u^t (Spalte 7). Sicherlich hängt es bei Kodierverfahren wie JPEG und GIF auch vom Bildinhalt ab, wieviele CPU-Zyklen benötigt werden. Beispielsweise war Bild *JJ07* zu 60% schwarz, wodurch weniger CPU-Zyklen benötigt wurden. Wie die Meßergebnisse zeigen, liegen die Werte jedoch dicht genug beieinander, um Rückschlüsse ziehen zu können. Dementsprechend kann für das Tiempo-Präsentationssystem die CPU-Menge zur Dekodierung eines Bildes als $C^o = Höhe \cdot Breite \cdot C_u^t$ berechnet werden.

Bild	Format	Volumen	Breite	Höhe	Zyklen	C_u^t
BD02	JPEG	129919	1020	728	470	6.3×10^{-4}
BD05	JPEG	22330	321	441	88	6.2×10^{-4}
BD11	JPEG	106328	800	545	279	6.4×10^{-4}
FF30	JPEG	140446	1088	750	516	6.3×10^{-4}
FF31	JPEG	173340	1050	735	498	6.5×10^{-4}
JC02	JPEG	42806	462	548	162	6.4×10^{-4}
JJ01	JPEG	23454	287	428	81	6.6×10^{-4}
JJ05	JPEG	25545	448	750	206	6.1×10^{-4}
JJ06	JPEG	59582	360	768	175	6.3×10^{-4}
JJ07	JPEG	57591	1024	768	422	5.3×10^{-4}
Star	JPEG	88285	561	768	276	6.4×10^{-4}
Jupiter	JPEG	54874	353	440	101	6.5×10^{-4}
Mars	GIF	87283	353	440	93	6.0×10^{-4}
SP07	GIF	386261	634	902	368	6.2×10^{-4}
SP08	GIF	178217	432	607	161	6.1×10^{-4}
ReRo	XPM	72488	227	149	6	1.8×10^{-4}
Rose	XPM	109927	227	149	7	2.0×10^{-4}

Tabelle 7-2: CPU-Zyklen zur Dekodierung von Bildern

Tabelle 7-3 zeigt die Messungen für die Dekodierung von HTML-Text. Wie man erkennt, gilt hier $C^o = Volumen \cdot C_u^t$.

Text	Volumen	Zyklen	C_u^t
BES0	800	7	8.8×10^{-3}
BES4	2319	21	9.0×10^{-3}
BES2	4545	40	8.8×10^{-3}
BES3	10570	92	8.7×10^{-3}
BES1	17591	156	8.8×10^{-3}

Tabelle 7-3: CPU-Zyklen zur Dekodierung von HTML-Text

Im Tiempo-Präsentationssystem ist ein Testbed integriert, das beim Starten des Präsentationssystems für jede Dekodierungs- und Darstellungskomponente anhand einer repräsentativen Menge kodierter Daten die Dekodierungskennzahl C_u^t berechnet. Die repräsentative Menge kodierter Daten wird bei den Messungen zum Finden der Formel für C^o bestimmt.

7.1.2 Zusammengesetzte Medien

Für ein zusammengesetztes Medium *o* müssen die Ressourcenanforderungen in Tabelle 7-4 bekannt sein.

Konstante	Semantik
C^o	Benötigte CPU-Zyklen in der Vorbereitungsphase
P^o	Benötigte Speichermenge während der Präsentationsphase

Tabelle 7-4: Ressourcenanforderungen zusammengesetzter Medien

Die Anzahl der in der Vorbereitungsphase benötigten CPU-Zyklen C^o hängt vom Typ des zusammengesetzten Mediums, der Implementierung der entsprechenden Darstellungskomponenten und der Leistungscharakteristik des Präsentationsterminals ab und kann mit Hilfe des Testbeds bestimmt werden. P^o wird vom Typ des Mediums und der Implementierung der ent-

sprechenden Darstellungskomponenten beeinflußt und ist somit ebenfalls systemabhängig. Betrachten wir beispielsweise ein zusammengesetztes Medium, das ein Fenster auf dem Bildschirm repräsentiert. Der CPU- und Speicherverbrauch dieses Mediums wird maßgeblich von den Datenstrukturen und Systemroutinen des Betriebssystems beeinflußt, mit Hilfe derer seine Semantik implementiert wird, und muß daher durch Messungen bestimmt werden.

7.1.3 Kontinuierliche Medien

Tabelle 7-5 zeigt die Ressourcenanforderungen, die für ein kontinuierliches Medium *o* bekannt sein müssen.

Konstante	Semantik
V^o	Größe einer kodierten Dateneinheit
C^o	Benötigte CPU-Zyklen zum Dekodieren und Ausspielen einer Dateneinheit
P^o	Benötigte Speichermenge zur Dekodierung der Mediendaten

Tabelle 7-5: Ressourcenanforderungen kontinuierlicher Medien

In Abhängigkeit der Kodierung eines kontinuierlichen Mediums können einzelne Dateneinheiten unterschiedlich groß sein. Beispielsweise bietet der Video-Kompressionsstandard MPEG [Gall91] die Möglichkeit, Bilder in Abhängigkeit von davor- und/oder dahinterliegenden Bildern zu kodieren. Die dadurch entstehenden sogenannten B- oder P-Frames sind kodiert im allgemeinen erheblich kleiner als I-Frames, die komplette Bilder repräsentieren. Abbildung 7-1 zeigt das mittlere Datenvolumen von jeweils zwanzig aufeinanderfolgender Frames zweier MPEG-Videos. *Video1* ist eine künstlich erzeugte Animation und *Video2* ein digitalisiertes Video. Wie zu sehen ist, differiert das Datenvolumen der Animation stärker als das des digitalisierten Videos. Das liegt daran, daß die meisten Werkzeuge zur Digitalisierung von aufgezeichneten Videos oder Audios so komprimieren können, daß ein Datenstrom mit einer relativ konstanten Datenrate entsteht. Es kann jedoch, wie das Beispiel von *Video1* zeigt, im allgemeinen nicht davon ausgegangen werden, daß dies so ist. Dementsprechend läßt sich nicht verhindern, daß man für Teilabschnitte von kontinuierlichen Medien, egal wie V^o gewählt wird, eine

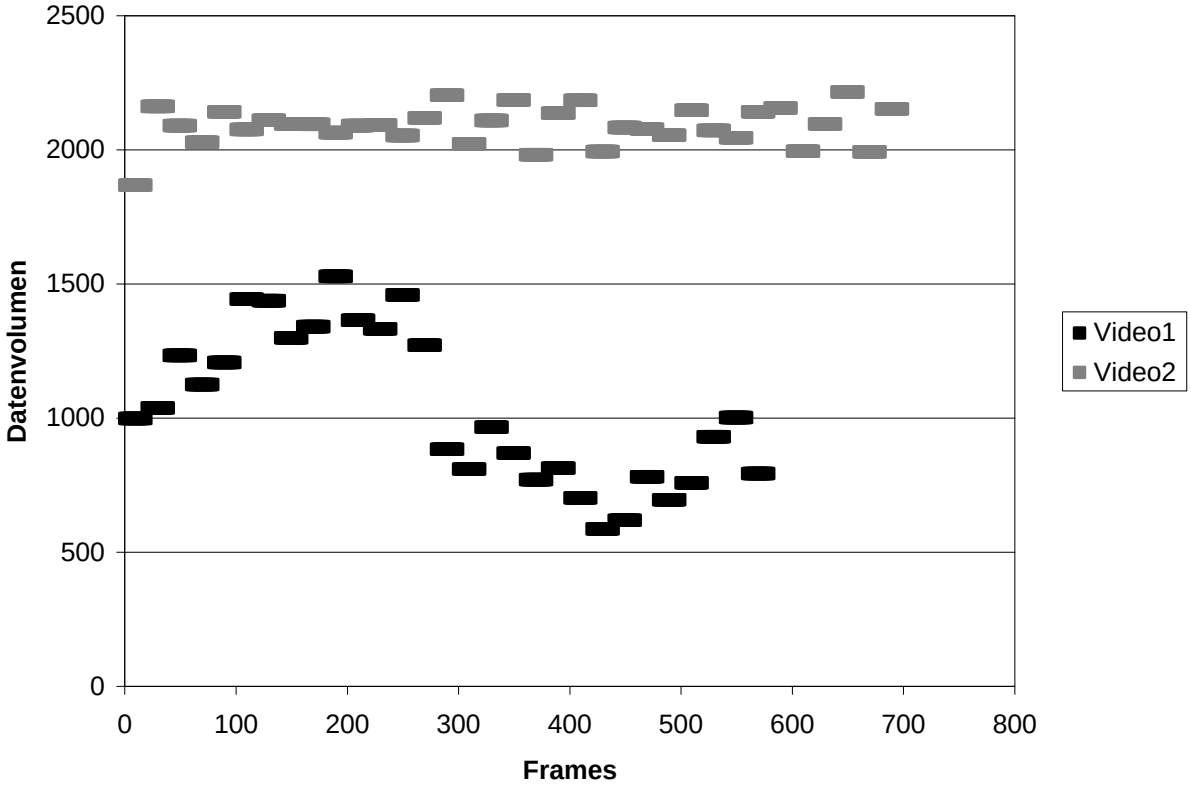


Abb. 7-1 : Datenraten von MPEG-Videos

zu große oder zu kleine Datenmenge beschreibt. Der Tiempo-Editor bestimmt V^o indem er die durchschnittliche Datenmenge in fünf Sekundenintervallen bestimmt, das Intervall mit der höchsten Datenmenge auswählt und diese durch die natürliche Präsentationsrate des Mediums dividiert. V^o wird dann in einem Attribut des Medienelements im Dokument abgelegt.

Um die benötigten CPU-Zyklen C^o zur Verarbeitung einer Dateneinheiten zu bestimmen, muß, wie bei statischen Medien, eine Formel durch Messungen gefunden werden, mittels derer anhand von Medieneigenschaften C^o berechnet werden kann. Abbildung 7-2 zeigt, wieviel CPU-Zyklen zur Verarbeitung eines Bildpunkts von *Video1* und *Video2* durch die Darstellungskomponente für Videos des Tiempo-Präsentationssystems benötigt werden. Beim Vergleich mit Abbildung 7-1 stellt man fest, daß die benötigten CPU-Zyklen relativ unabhängig vom Volumen der Frames sind. Somit hängen die für die Verarbeitung eines Frames benötigten CPU-Zyklen nahezu ausschließlich von der Bildgröße ab. Dementsprechend ist es auch bei kontinuierlichen Medien möglich, eine Dekodierungskennzahl C_u^t mit Hilfe eines Testbeds anhand einer repräsentativen Datenmenge und C^o anhand einer entsprechenden Formel zu bestimmen.

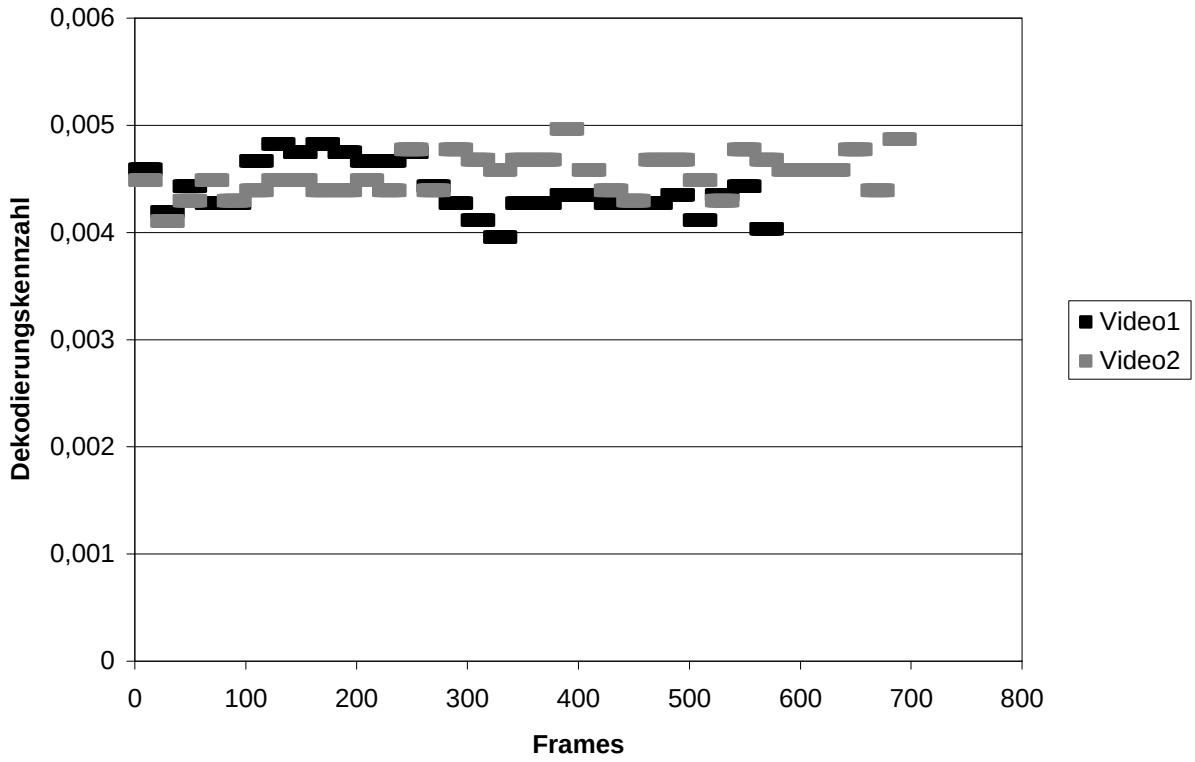


Abb. 7-2 : Dekodierungskennzahlen für MPEG-Videos

P^O läßt sich im Prinzip wie bei diskreten Medien anhand der Eigenschaften des Mediums und der Implementierung der Dekodierungs- und Darstellungskomponenten bestimmen. In Abhängigkeit des verwendeten Kodierungsverfahrens kann es hierbei erforderlich sein, mehrere hintereinander auszuspielende Dateneinheiten in dekodierter Form zu puffern, wie beispielsweise bei MPEG-kodierten Videos. Hier muß immer das letzte Vollbild (I-Frame) gespeichert werden, so daß darauffolgende P- und B-Frames dekodiert werden können.

7.1.4 Messung positionsunabhängiger Ressourcenanforderungen

Während der Vorbereitungsphase vom Medienobjekten mißt das Tiempo-Präsentationssystem laufend die für ein Medium transferierte Datenmenge und die verbrauchten CPU-Zyklen. Diese Informationen werden bei der Berechnung von Adaptionen benötigt, um die noch ausstehenden Ressourcenanforderungen nicht abgeschlossener Vorbereitungsaktionen bestimmen zu können.

7.2 Vorhersage verfügbarer Ressourcen

Die Vorhersage, der im weiteren Verlauf der Präsentation verfügbaren Ressourcen, basiert auf der Beobachtung verfügbarer Ressourcen während der bisherigen Präsentation. Es müssen Vorhersagen für den verfügbaren Speicher, die verfügbaren CPU-Zyklen, die verfügbare Bandbreite sowie die Kommunikationsverzögerung zwischen Servern und dem Präsentationsterminal gemacht werden.

7.2.1 Bestimmung der aktuell verfügbaren Ressourcen

Die momentan verfügbaren Ressourcen werden durch Beobachtung erfaßt. Die Menge freien Speichers P_f des Präsentationsterminals wird vom Betriebssystem erfragt. Die Menge freien Speichers, der für die Präsentation zur Verfügung steht, berechnet sich dann als

$$P_a = P_f + \sum_o P^o - Q_P \qquad (7\text{-}1)$$

Zur Menge des nicht verwendeten Speichers wird der Speicher hinzugezählt, der momentan durch die Medienobjekte *o* der Dokumentpräsentation verbraucht wird. Dadurch ergibt sich die Menge Speicher P_a, die für die Präsentation verplant werden kann. Q_P $(Q_P \geq 0)$ ist ein Qualitätsparameter, der beschreibt, wieviel des verfügbaren Speichers nicht für die Präsentation verplant wird. Dieser Parameter hilft, Ungenauigkeiten bei der Bestimmung des Speicherbedarfs von Medien auszugleichen und schafft einen gewissen Puffer für plötzlich auftretende Speicherallokationen durch konkurrierende Anwendungen.

Die Menge freier CPU-Zyklen des Präsentationsterminals C_f wird vom Betriebssystem erfragt. Die Menge verfügbarer CPU-Zyklen, die für die Präsentation zur Verfügung steht, berechnet sich äquivalent zur Menge des verfügbaren Speichers als

$$C_a = C_f + \sum_o C^o - Q_C \qquad (7\text{-}2)$$

D.h. auch hier werden zur Menge der nicht verwendeten CPU-Zyklen die CPU-Zyklen hinzugezählt, die momentan durch die Medienobjekte *o* der Präsentation verbraucht werden. Q_C $(Q_C \geq 0)$ ist ein Qualitätsparameter, der beschreibt, wieviel CPU-Zyklen nicht für die Prä-

sentation verplant werden. Dieser Parameter hilft Ungenauigkeiten bei der Bestimmung des CPU-Verbrauchs von Medienobjekten auszugleichen.

Um die Bandbreite zwischen einem Server *y* und dem Präsentationsterminal zu bestimmen, wird – wenn Daten transferiert werden – beobachtet, welches Datenvolumen B_r^y pro Zeiteinheit auf dem Präsentationsterminal empfangen wird. Anhand des Ablaufplans der Präsentation ist bekannt, wieviele Daten pro Zeiteinheit von einem Server transferiert werden sollten: $\sum_o B^o$. Die verfügbare Bandbreite B_a^y bestimmt sich dann als

$$B_a^y = \begin{cases} B_r^y - Q_B & \text{falls} \quad B_r^y < \sum_o B^o \\ B_r^y + b - Q_B & \text{sonst} \end{cases} \tag{7-3}$$

Falls alle Daten wie geplant transferiert wurden, nehmen wir an, daß noch um *b* mehr Bandbreite zur Verfügung steht. Falls weniger Daten als geplant transferiert wurden, wissen wir aufgrund der empfangenen Datenmenge wie groß die Bandbreite momentan ist. Q_B $(Q_B \geq 0)$ ist ein Qualitätsparameter, der beschreibt, wieviel Bandbreite nicht für die Präsentation verplant wird. Dieser Parameter hilft Ungenauigkeiten bei der Bestimmung des Bandbreitenbedarfs von Medienobjekten auszugleichen und puffert Beeinträchtigungen durch konkurrierende Anwendungen.

Des weiteren muß die Antwortzeit von Servern *y* bekannt sein. Die Antwortzeit d_r^y ist die Summe der zweifachen Laufzeit von Nachrichten d_c^y und der Reaktionszeit des Servers d_s^y:

$$d_r^y = 2d_c^y + d_s^y \tag{7-4}$$

Auf dem Präsentationsterminal kann lediglich die Antwortzeit d_r^y gemessen werden. Die Aufteilung in Laufzeit und Reaktionszeit ist ohne zusätzliche Messungen nicht zu bestimmen, wird aber auch nicht benötigt.

Um die Puffergröße beim Stromtransfer kontinuierlicher Medien bestimmen zu können, wird die Varianz j^y der Kommunikationsverzögerungen (Jitter) jedes Servers *y* anhand der Empfangszeitpunkte von Dateneinheiten kontinuierlicher Medien berechnet.

Bei der Bestimmung der Werte P_f, C_f, B_r^y, d_r^y und j^y empfielt sich das Glätten der Meßwerte durch Herausfiltern kurzfristiger Schwankungen, da sonst häufig überflüssige Adaptionen ausgelöst werden, wodurch schnell ein Oszillieren entstehen kann. Hierzu kann beispielsweise ein geometrischer Filter der Form

$$R_a^i = gR_a^{i-1} + (1-g)R_a, \qquad R_a^0 = R_a, \qquad 0 \le g \le 1 \tag{7-5}$$

eingesetzt werden, der bei der Bestimmung der momentan verfügbaren Menge einer Ressource R_a^i sowohl den aktuelle Meßwert als auch die vorherigen Meßwerte berücksichtigt. Je nach Wahl der Konstante *g* verstärkt sich der Einfluß des neuen Meßwerts R_a oder der Einfluß der akkumulierten alten Meßwerte R_a^{i-1}.

7.2.2 Interpolation verfügbarer Ressourcen

Mit Hilfe der Informationen über die momentan verfügbaren Ressourcen muß eine Prognose für die Ressourcensituation im weiteren Verlauf der Präsentation erstellt werden. Hierbei gibt es zwei grundsätzliche Möglichkeiten.

7.2.2.1 Die Historie über verfügbare Ressourcen wird nicht berücksichtigt

Bei diesem Ansatz wird angenommen, daß zu jedem Zeitpunkt in der Zukunft genauso viele Ressourcen verfügbar sein werden, wie dies momentan der Fall ist. Die Historie über verfügbare Ressourcen wird also nicht berücksichtigt. Dadurch ergibt sich eine konstante Prognosefunktion $R_a(t)$, die beschreibt wieviel Ressourcen zu jedem Zeitpunkt *t* in der Zukunft verfügbar sein werden:

$$R_a(t) = R_a \tag{7-6}$$

7.2.2.2 Die Historie über verfügbare Ressourcen wird berücksichtigt

Bei diesem Ansatz wird angenommen, daß die Ressourcensituation in der Zukunft ähnlich sein wird, wie sie in der Vergangenheit war. D.h. es werden auch Informationen über verfügbare Ressourcen in der Vergangenheit berücksichtigt. Dieser Ansatz scheint vielversprechender zu

sein als der erste, da hierbei mehr Informationen berücksichtigt werden und somit vermutlich eine bessere Vorhersage entsteht. Da sich die Zukunft jedoch völlig anders als die Vergangenheit verhalten kann, ist eine detaillierte Nachbildung der Vergangenheit nicht sinnvoll. Man beschränkt sich besser darauf, lediglich die Tendenz zu beschreiben. Dies kann wie folgt geschehen.

Aufgrund der Beobachtungen ist die Menge R_a der aktuell (d.h. zum Zeitpunkt t_c) verfügbaren Ressourcen bekannt. Es wird nun ein Wert R_m bestimmt, der angibt, wieviel Ressourcen in der Vergangenheit verfügbar waren. R_m kann beispielsweise mit Hilfe eines geometrischen Filters der Form (7-5) aus den gefilterten Werten R_a berechnet werden. Je nach Wahl des Parameters g wird hierbei mehr oder weniger Historie berücksichtigt. Der Wert R_m beschreibt somit die längerfristige Tendenz der Ressourcensituation in der Vergangenheit. Nimmt man nun an, daß sich die Ressourcensituation in der Zukunft so ähnlich gestalten wird, wie sie in der Vergangenheit war, kann eine Prognose auf folgende Weise erstellt werden.

Es wird eine Funktion $R_a(t)$ konstruiert, die prognostiziert wieviel Ressourcen zu jedem Zeitpunkt t in der Zukunft verfügbar sein werden. Aus rechentechnischen Gründen sollte keine allzu komplexe Kurve gewählt werden. Es bieten sich somit lineare oder hyperbolische Kurven an.

Eine lineare Funktion kann wie folgt konstruiert werden:

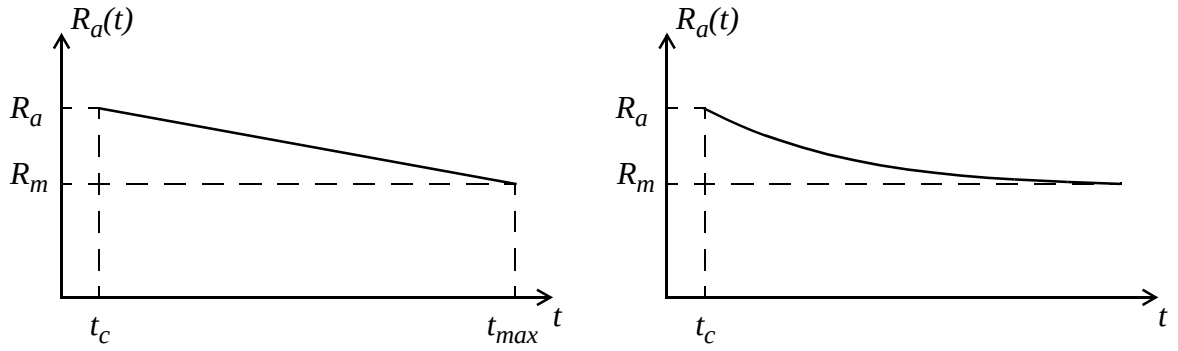
$$R_a(t) = \frac{R_m - R_a}{t_{max} - t_c}(t - t_c) + R_a \qquad (7\text{-}7)$$

Diese Funktion (siehe Abbildung 7-3a) hat für den Zeitpunkt t_c den Funktionswert R_a und verläuft dahinter linear bis zum Zeitpunkt t_{max}, an dem sie den Funktionswert R_m liefert. t_{max} ist ein Zeitpunkt nach dem Ende der Präsentation.

Eine hyperbolische Funktion kann auf folgende Art konstruiert werden:

$$R_a(t) = \frac{\text{sign}(R_a - R_m)(R_a - R_m)^2}{t - t_c + |R_a - R_m|} + R_m \qquad (7\text{-}8)$$

Bei einer solchen Funktion (siehe Abbildung 7-3b) liegt der Scheitelpunkt der Kurve im aktuellen Zeitpunkt t_c und hat den Funktionswert R_a, dahinter nähert sich die Kurve an R_m an.



a) Lineare Interpolation b) Hyperbolische Interpolation

Abb. 7-3 : Prognose verfügbarer Ressourcen

Ein Vergleich der beiden Ansätze ergibt, daß der lineare Ansatz aus rechentechnischen Gesichtspunkten einfacher ist. Der hyperbolische Ansatz hat hingegen den Vorteil, daß sich die Kurve schneller an R_m annähert, was bei langen Präsentationen zu besseren Vorhersagen führen könnte.

7.2.3 Evaluierung der Ansätze

Beide vorgestellten Ansätze wurden im Rahmen des Tiempo-Präsentationssystems implementiert und getestet. Dabei konnte nicht festgestellt werden, daß einer der Prognoseansätze deutlich besser als der andere ist. Da der Ansatz, bei dem die Historie über verfügbare Ressourcen nicht verwendet wird, rechentechnische Vorteile bei der Ablaufplanung bringt, wurde letztendlich er gewählt.

8 Das adaptive Tiempo-Ablaufplanungsverfahren

In diesem Abschnitt wird das Tiempo-Ablaufplanungsverfahren [Wira99c] vorgestellt, das zur Klasse der proaktiven adaptiven Ablaufplanungsverfahren gehört. Das Tiempo-Ablaufplanungsverfahren berücksichtigt bei der Erstellung eines Ablaufplans nicht nur die Ressourcensituation, sondern auch die Prioritäten, die Präsentationsalternativen zugewiesen sind. Dementsprechend ist die Ablaufplanung ein Optimierungsproblem.

Eine geschlossene mathematische Lösung eines solchen Optimierungsproblems erfordert, daß alle Nebenbedingungen, wie Ressourceneinschränkungen und die temporale Spezifikation, durch Gleichungen beschrieben werden. Ein Modell für diesen Ansatz wurde in [Wira99b] vorgestellt. Problematisch an diesem Ansatz ist, daß sehr schnell sehr große mathematische Optimierungssysteme (Mathematische Programme) entstehen, deren Lösung aufwendig ist. Im folgenden wird daher ein optimierendes Ablaufplanungsverfahren vorgestellt, das ohne die explizite Erzeugung von Mathematischen Programmen auskommt.

In Abschnitt 8.1 wird das zur Ablaufplanung verwendete Datenmodell vorgestellt. In den darauffolgenden Abschnitten wird erörtert, wie ein Ablaufplan berechnet und adaptiert wird. In Abschnitt 8.6 wird erläutert, wie eine Präsentation anhand eines Ablaufplans gesteuert wird. Schließlich wird das vorgeschlagene Ablaufplanungsverfahren evaluiert.

8.1 Generierung eindeutiger Ereignisgraphen

Die Einplanung des Ressourcenbedarfs von Präsentationsaktionen der Präsentationsphasen läßt sich nur dann einigermaßen effizient durchführen, wenn sich diese eindeutig überlappen. Für Präsentationsaktionen der Vorbereitungsphasen ist dies nicht erforderlich. Bevor die eigentliche Ablaufplanung beginnt, wird eine Dokumentspezifikation daher in sogenannte **eindeutige Ereignisgraphen** übersetzt. Ein solcher Graph zeichnet sich dadurch aus, daß er eine eindeutige Überlappung der Präsentationsphasen von Projektoren beschreibt, und damit auch eine eindeutige Überlappung der Aktionen dieser Präsentationsphasen impliziert.

8.1.1 Bestimmung von Präsentationsalternativen

Der erste Schritt bei der Bestimmung eines Ablaufplans besteht darin, alle möglichen Kombinationen von Präsentationsalternativen aus Auswahlgruppen in einem Dokument zu bestimmen. Dies erfolgt mit Hilfe von Alternativenbäumen gemäß dem in Abschnitt 5.5 beschriebenen Verfahren. In Abbildung 8-1 ist eine einfache Dokumentspezifikation zu sehen, die eine Auswahlgruppe mit zwei Präsentationsalternativen enthält. Dementsprechend würde man für dieses Dokument zwei Kombinationen erhalten. Anschließend werden für jede Kombination die Prioritäten der enthaltenen Präsentationsalternativen addiert. Der dabei errechnete Wert beschreibt die **Dienstgüte der Kombination**. Für das Beispieldokument hat die Präsentationsalternativenkombination, welche *VideoB* enthält, die Priorität 100, und die Kombination, welche das *Bild* enthält, die Priorität 20.

8.1.2 Erstellung eindeutiger Ereignisgraphen

Für jede Kombination von Präsentationsalternativen wird ein eindeutiger Ereignisgraph erstellt. Dazu werden die in der betrachteten Kombination enthaltenen Präsentationsalternativen als aktiv markiert und ebenso alle nicht in Präsentationsalternativen enthaltene Dokumentelemente. Die Menge aktiver Dokumentelemente kann durch Dienstgütebereiche definierte Flexibilität enthalten. Um die zeitlichen Abhängigkeiten der aktiven Dokumentelemente darzustellen, werden, wie in Abschnitt 5.2.3 beschrieben, Ereignisgraphen erzeugt. Enthält die Dokumentspezifikation keine indeterministischen Multimedia-Projektoren, entsteht nur ein Ereignisgraph pro Kombination, ansonsten entstehen mehrere Ereignisgraphen. Alle aus einer Kombination entstehenden Ereignisgraphen implizieren die gleiche Dienstgüte. Eindeutige Ereignisgraphen werden aus den auf diese Weise generierten Ereignisgraphen G_i mit Hilfe des folgenden Algorithmus erzeugt.

8.1.2.1 Bestimmung der vollständig konsistenten Kantenlängen

Zur Reduzierung des Aufwands werden Knoten, die durch Kanten der Länge Null verbunden sind, zusammengefaßt und durch einen **zusammengesetzten Knoten** repräsentiert. Durch Anwendung des Längenbestimmungsalgorithmuses aus Abschnitt 5.2.4 werden die möglichen Abstände zwischen allen Knoten eines Ereignisgraphs G_i bestimmt. Abbildung 8-2 zeigt den

```
<Document  Id="SimpleDoc" >
        <Projection  PresInterval="90" Speed="1.0" Rate="1.0" >
                <ForceEnd  Instant="0" Position="first" />
        </Projection>
        <VideoPro  Id="VideoA" >
                <Projection  PresInterval="[0:0,?:0]" Speed="2.0" Rate="[0.6:0,1:50]:d" >
                        <AlignLength/>
                </Projection>
                <Video  Length="100" .../>
        </VideoPro>
        <VideoPro  Id="VideoB" >
                <Projection  PresInterval="[0:0,?:0]" Speed="1.0" Rate="[0.8:0,1:80]:d" >
                        <AlignLength/>
                </Projection>
                <Video  Length="60" .../>
        </VideoPro>
        <PicturePro  Id="Bild" >
                <Projection  PresInterval="[40:0,60:50]" Speed="0" Rate="1.0" >
                        <HoldData  Instant="0" Position="first" />
                </Projection>
                <Picture .../>
        </PicturePro>
        <ButtonPro  Id="Knopf">
                <Projection  PresInterval="[0:0,?:0]" Speed="0" Rate="1.0" >
                        <HoldData  Instant="0" Position="first" />
                </Projection>
                <ButtonIntMng  Id="KnopfM" IntInterval="[0:0,?:0]"
                                StartDelay="0" EndDelay="0" State="released" />
                <Picture .../>
        </ButtonPro>
        <Delayed  Id="DelayA" From="VideoA" To="VideoB"
                        Delay1="[40:60,60:0]" Delay2="[0:0,?:0]" />
        <Before  Id="BeforeA" From="VideoA" To="Bild" Delay1="[0:50,5:0]" />
        <While  Id="WhileA" From="SimpleDoc" To="Knopf" Delay1="0" Delay2="0" />
        <Reaction  Id="ReactionA" IntManager="KnopfM"
                        PreCondition="State='pressed'" PostCondition="State='released'" >
                <JumpTo  Element="SimpleDoc" Instant="0" />
        </Reaction>
        <SelectionGroup  Id="SelectionGroupA" Policy="dynamic" >
                <Alternative  Priority="100" Elements="VideoB" />
                <Alternative  Priority="20" Elements="Bild" />
        </SelectionGroup>
</Document>
```

Abb. 8-1 : Beispieldokument

auf diese Art entstehenden Ereignisgraph für die Präsentationsalternativenkombination mit *VideoB* des Dokuments in Abbildung 8-1.

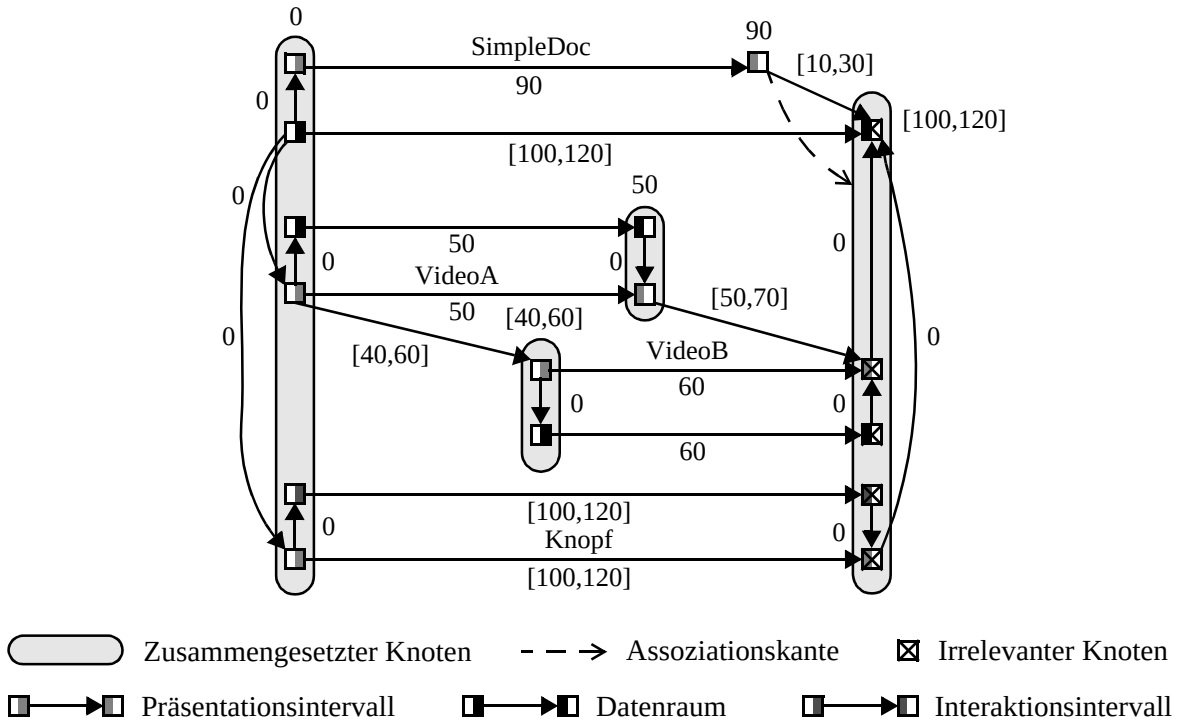


Abb. 8-2 : Ereignisgraph Video-Präsentationsalternativenkombination

8.1.2.2 Markierung irrelevanter Knoten

Die Präsentationsphase eines Projektors wird durch sein Präsentationsintervall beschrieben. Der Teil der Präsentationsphase eines in einem Multimedia-Projektor enthaltenen Projektors, der zeitlich gesehen nicht in der Präsentationsphase des Multimedia-Projektors liegt, wird nicht präsentiert und kann sich damit auch nicht mit den Präsentationsphasen anderer Projektoren überlappen. Somit können Knoten, die nicht präsentierte Teile von Präsentationsphasen begrenzen, als irrelevant markiert werden, da sie sich beliebig verschieben dürfen.

Dazu wird für jeden Projektor *o* die Menge I_o der Projektoren bestimmt, die in *o* enthalten sind. Es sei K_o die Knotenmenge, welche die Elemente in I_o im Ereignisgraph repräsentiert, erweitert um die Datenraumknoten von *o*. Ein Knoten in K_o ist irrelevant, wenn er im Ereignisgraph immer vor dem Anfangsknoten des Präsentationsintervalls von *o* liegt oder immer hinter dem Endknoten des Präsentationsintervalls von *o* liegt. Ob ein Knoten diese Bedingungen erfüllt,

kann anhand der berechneten Längenmatrix und der Kanten im Ereignisgraph geprüft werden. Die Funktion irrelevanter Knoten übernimmt der entsprechende Anfangs- oder Endknoten des Präsentationsintervalls des umgebenden Multimedia-Projektors. Um diesen Sachverhalt zu beschreiben, werden sogenannte **Assoziationskanten** in den Graph eingefügt. In Beispiel 8-2 übernimmt der Endknoten des Präsentationsintervalls des Dokuments *SimpleDoc* die Funktion der als irrelevant markierten Knoten.

8.1.2.3 Bestimmung der Start- und Stoppknoten von Präsentationsphasen

Entsprechend der spezifizierten Ausrichtungspolitik wird für jeden Projektor festgelegt, wann er wie Dateneinheiten ausspielt und damit auch, wann welche Ressourcen benötigt werden. Hierbei unterscheidet man **diskrete Projektoren**, die diskrete Medien präsentieren, **kontinuierliche Projektoren**, die kontinuierliche Medien präsentieren, und **Multimedia-Projektoren**.

Für einen diskreten Projektor *o* wird, wie in Abbildung 8-3 dargestellt, zum Zeitpunkt, den der Präsentationsintervallanfangsknoten repräsentiert, mit dem Ausspielen der Dateneinheit begonnen und zum Zeitpunkt, den der Präsentationsintervallendknoten repräsentiert, wird das Ausspielen beendet. Der Präsentationsintervallanfangsknoten wird daher als **Startknoten** $S^o_{D^1}$ und der Präsentationsintervallendknoten als **Stoppknoten** $E^o_{D^1}$ einer diskreten Präsentationsphase D_1 bezeichnet.

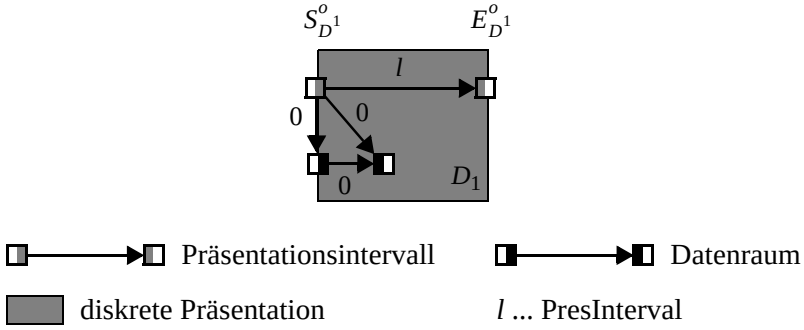


Abb. 8-3 : Verhalten diskreter Projektoren

Ein kontinuierlicher Projektor *o* für den die Ausrichtungspolitik *HoldData* spezifiziert wurde, impliziert eine kontinuierliche Präsentationsphase *K* und, wie in Abbildung 8-4 dargestellt, in Abhängigkeit der Anordnung der zugehörigen Präsentationsintervall- und Datenraumknoten im

Ereignisgraph teilweise bis zu zwei diskrete Präsentationsphasen D_1, D_2. Der Anfangsknoten des Präsentationsintervalls oder Datenraums bei dem K beginnt, wird als Startknoten S^o_K bezeichnet und der Endknoten des Präsentationsintervalls oder Datenraums bei dem K endet, wird als Stoppknoten E^o_K bezeichnet. Existiert D_1, ist der Präsentationsintervallanfangsknoten der Startknoten $S^o_{D^1}$ und der Datenraumanfangsknoten der Endknoten $E^o_{D^1}$. Existiert D_2, ist der Datenraumendknoten der Startknoten $S^o_{D^2}$ und der Präsentationsintervallendknoten der Endknoten $E^o_{D^2}$.

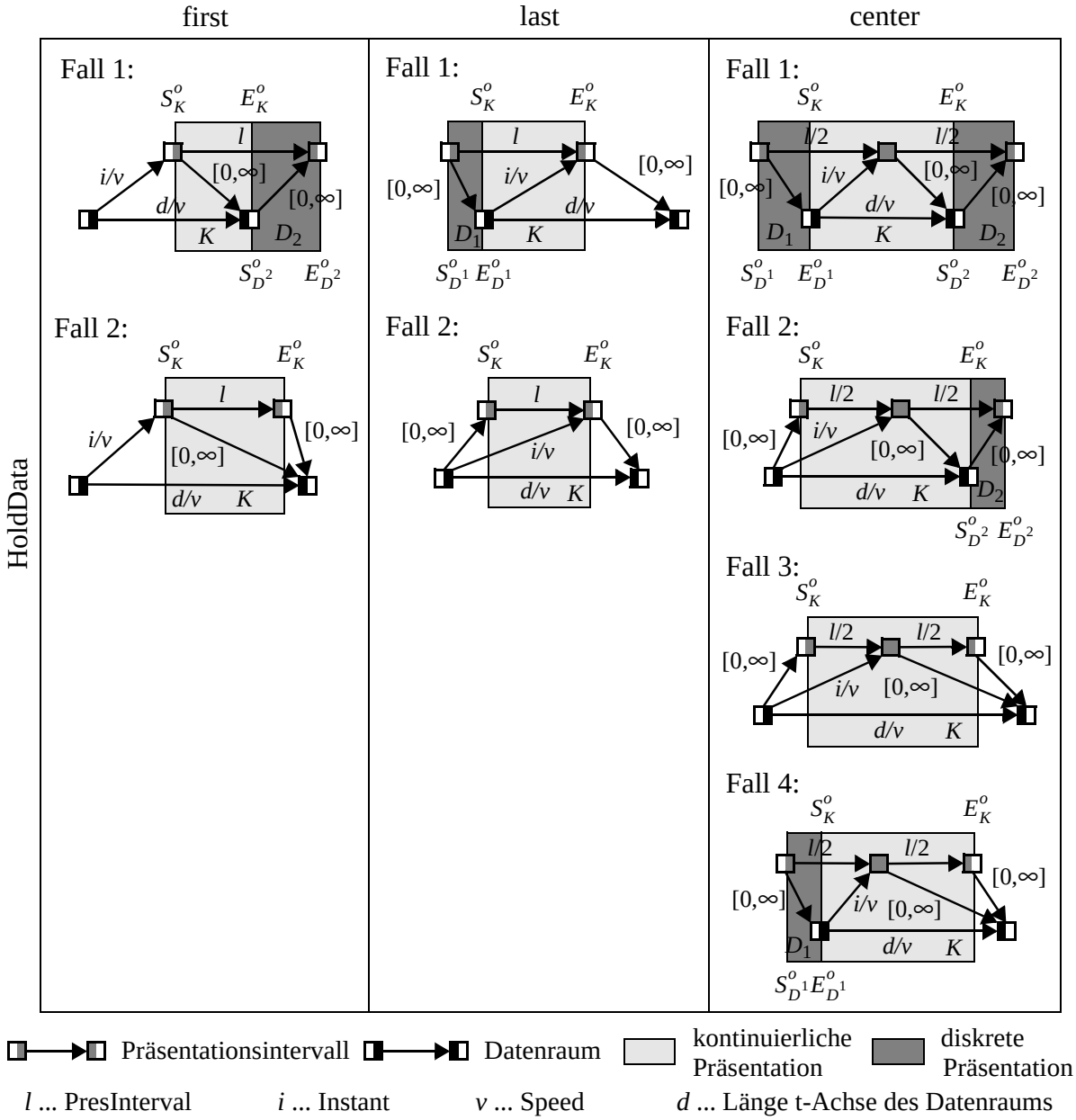


Abb. 8-4 : Verhalten kontinuierlicher Projektoren bei der Ausrichtungspolitik *HoldData*

Bei allen anderen Ausrichtungspolitiken implizieren kontinuierliche Projektoren, wie in Abbildung 8-5 dargestellt, nur eine kontinuierliche Präsentationsphase *K*. Hierbei ist der Präsentationsintervallanfangsknoten der Startknoten S_K^o und der Präsentationsintervallendknoten der Stoppknoten E_K^o.

Bei Multimedia-Projektoren können – vergleichbar mit kontinuierlichen Projektoren – bei der Ausrichtungspolitik *HoldData* unterschiedliche Präsentationsphasen auftreten. Eine Präsentationsphase K^M kann mit der *K*-Phase eines kontinuierlichen Projektors (siehe Abbildung 8-4) verglichen werden. Die Präsentationsphasen D_1^M und D_2^M entsprechen den D_1- und D_2-Phasen kontinuierlicher Projektoren.

In einer D_1-Phase wird, laut Definition der Ausrichtungspolitik *HoldData*, der erste Zeitpunkt des Datenraums ausgespielt, in einer D_2-Phase der letzte Zeitpunkt des Datenraums. Diese Semantik wird für Multimedia-Projektoren wie folgt umgesetzt.

Existiert für einen Multimedia-Projektor eine D_1^M-Phase, ist der Startknoten dieser Phase auch der Startknoten $S_{D^1}^o$ einer D_1-Phase der Projektoren deren Präsentation am Anfang des Multimedia-Datenraums beginnt. Ist ein solcher Projektor zu diesem Zeitpunkt in keiner D_1-Phase, ist der Datenraumanfangsknoten des Multimedia-Projektors der Stoppknoten $E_{D^1}^o$ des Projektors. Anderenfalls besitzt der Projektor bereits einen $E_{D^1}^o$-Knoten.

Existiert für einen Multimedia-Projektor eine D_2^M-Phase, ist der Stoppknoten der D_2^M-Phase auch der Stoppknoten $E_{D^2}^o$ einer D_2-Phase der Projektoren deren Präsentation am Ende des Multimedia-Datenraums endet. Ist ein solcher Projektor zu diesem Zeitpunkt in keiner D_2-Phase, ist der Datenraumendknoten des Multimedia-Projektors der Startknoten $S_{D^2}^o$ des Projektors. Ansonsten besitzt der Projektor bereits einen $S_{D^2}^o$-Knoten.

Wie in Abbildung 8-5 zu sehen ist, gibt es bei den anderen Ausrichtungspolitiken keine D_1^M- oder D_2^M-Phasen. Dementsprechend sind auch keine Start- oder Stoppknoten zu verschieben. Bei der Ausrichtungspolitik *LoopData* werden auftretende Lücken durch Kopien des Datenrauminhalts des Multimedia-Projektors gefüllt.

Für einen Multimedia-Projektor werden keine Start- und Stoppknoten für K^M-, D_1^M- und D_2^M-Phasen identifiziert, da der Ressourcenbedarf eines Multimedia-Projektors, wie in Abschnitt 6

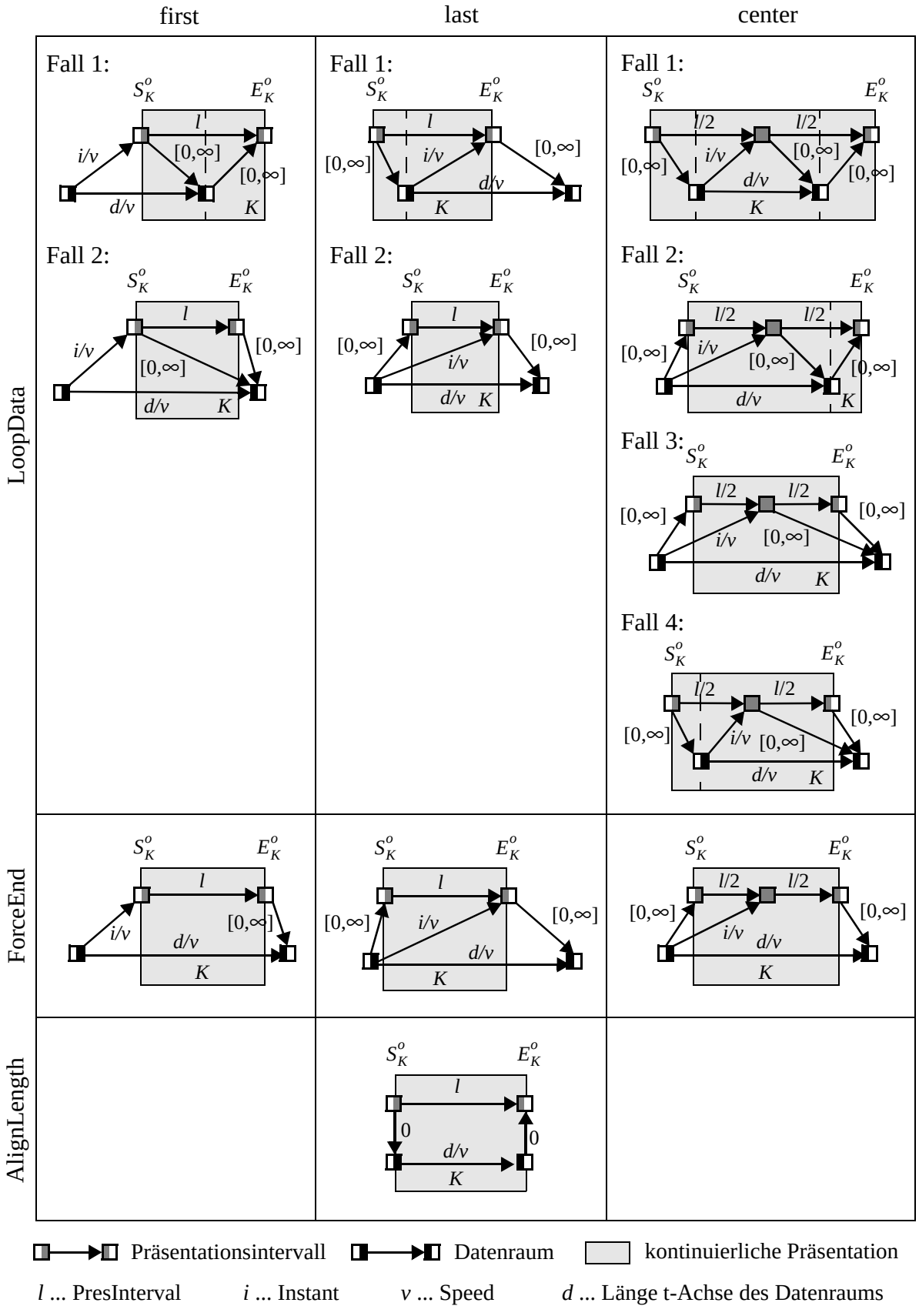


Abb. 8-5 : Verhalten kontinuierlicher Projektoren in Abhängigkeit der Ausrichtungspolitik

gezeigt wurde, in der Präsentationsphase konstant ist. Es wird lediglich der Präsentationsintervallanfangsknoten als Startknoten S_M^o und der Präsentationsintervallendknoten als Stoppknoten E_M^o markiert.

Die Start- und Stoppknoten der Projektoren werden bestimmt, indem man einen Ereignisgraph durchläuft und anhand der relativen Lage der Knoten, wie oben beschrieben, entscheidet, welche Funktion ein relevanter Knoten hat. Hierbei werden die funktionalen Verschiebungen berücksichtigt, die durch Assoziationskanten beschrieben sind. Existiert eine Präsentationsphase aufgrund von funktionalen Verschiebungen nicht, erkennt man dies daran, daß ihre Start- und Stoppknoten durch Assoziationskanten mit demselben Knoten verbunden sind. Sind alle Start- und Stoppknoten bestimmt, ist der zeitliche Verlauf der Ressourcenanforderungen der Projektoren ableitbar. Abbildung 8-6 zeigt die Start- und Stoppknoten des Beispiels in Abbildung 8-2.

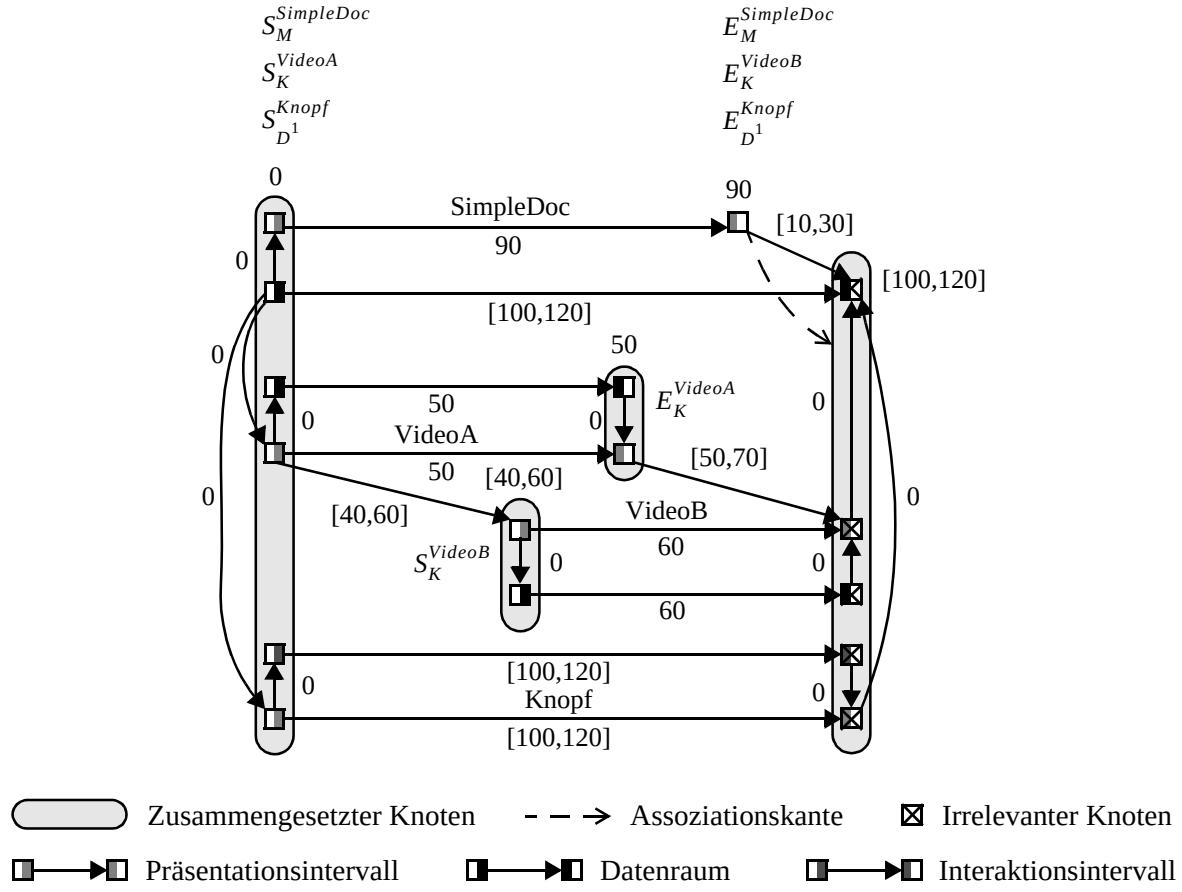


Abb. 8-6 : Bestimmung von Start- und Stoppknoten in Ereignisgraphen

8.1.2.4 Einplanungstest für positionsabhängige Ressourcenanforderungen

Um Ereignisgraphen, die einen zu hohen Ressourcenverbrauch implizieren, frühzeitig auszusortieren, wird überprüft, ob positionsabhängige Ressourcenanforderungen potentiell erfüllbar sind.

Es sei $x_j = [\underline{x}_j, \bar{x}_j]$ das Zeitintervall, genannt **Ereignisintervall**, in dem das mit einem Knoten *j* assoziierte Ereignis eintreten kann. Ereignisintervalle können aus der Längenmatrix abgelesen werden. Die positionsabhängigen Ressourcenanforderungen für eine Ressource *R* sind potentiell erfüllbar, wenn für jedes x_j eines Start- oder Stoppknotens die folgenden Restriktionen erfüllt sind:

$$\begin{aligned} \sum_{\left\{o:\left(\left(\bar{x}_{S_p^o} < \underline{x}_j\right) \wedge \left(\bar{x}_j < \underline{x}_{E_p^o}\right)\right) \vee \left(j = E_p^o\right)\right\}} \underline{R}^o(x_j) &\le \bar{R}_a(x_j) \\ \sum_{\left\{o:\left(\left(\bar{x}_{S_p^o} < \underline{x}_j\right) \wedge \left(\bar{x}_j < \underline{x}_{E_p^o}\right)\right) \vee \left(j = S_p^o\right)\right\}} \underline{R}^o(x_j) &\le \bar{R}_a(x_j) \end{aligned} \qquad (8\text{-}1)$$

In den Ungleichungen (8-1) repräsentiert $\bar{R}_a(x_j)$ die maximale Anzahl der in einem Zeitpunkt des Zeitintervalls x_j verfügbaren Einheiten der Ressource *R*. Die Funktion $\underline{R}^o(x_j)$ beschreibt die minimale Menge der positionsabhängigen Ressource *R*, die ein Projektor *o* im Zeitintervall x_j benötigt. $\underline{R}^o(x_j)$ ist damit ein Platzhalter für die Funktionen (6-3), (6-8), (6-12) und (6-13), die beschreiben wieviel positionsabhängige CPU-Zyklen, Bandbreite und Speicher ein Projektor zu jedem Zeitpunkt benötigt. Die minimale positionsabhängige Ressourcenanforderung ergibt sich, wenn der Funktionswert (6-8), (6-12) und (6-13) für die minimale Präsentationsrate berechnet wird.

Bei jedem x_j kann die Präsentation einiger Projektoren enden und die anderer Projektoren starten, wodurch der Ressourcenbedarf vor und hinter einem Ereignisintervall x_j unterschiedlich sein kann. Die erste Ungleichung (8-1) berücksichtigt beim Aufsummieren der Ressourcenanforderungen keine Projektoren, die im Zeitintervall x_j mit ihrer Präsentations starten, und die zweite Ungleichung (8-1) berücksichtigt beim Aufsummieren der Ressourcenanforderungen keine Projektoren, die im Zeitintervall x_j ihre Präsentation beenden. Somit prüft die erste Ungleichung die Ressourcensituation vor x_j und die zweite Ungleichung hinter x_j. Es ist ausrei-

chend die Ressourcensituation in den Zeitintervallen x_j zu prüfen, da aus Präsentationssicht nur in diesen Intervallen eine Veränderung der Ressourcenanforderungen eintritt und die Funktionen $R_a(t)$ monoton sind.

Ist eine der Ungleichungen (8-1) nicht erfüllt, sind zu wenig Ressourcen vorhanden und der Ereignisgraph kann gelöscht werden.

8.1.2.5 Bestimmung eines Knotenpaars mit verschiedener Überlappung

Es wird ein Paar relevanter Knoten (*A*, *B*) gesucht, deren minimaler Abstand in der Längenmatrix positiv ist und deren maximaler Abstand in der Längenmatrix positiv ist. Denn genau dann kann zeitlich gesehen Ereignis *A* vor oder hinter Ereignis *B* liegen. Interaktionsintervallknoten spielen hierbei keine Rolle. In Abbildung 8-6 erfüllt das Knotenpaar Präsentationsintervallendknoten *VideoA* und Präsentationsintervallanfangsknoten *VideoB* diese Bedingung.

Ist kein Knotenpaar vorhanden, das diese Bedingung erfüllt, ist der Ereignisgraph ein eindeutiger Ereignisgraph. Ist ein solches Knotenpaar gefunden worden, wird der in Abschnitt 8.1.2.6 beschriebene Schritt durchgeführt.

8.1.2.6 Erzwingen einer eindeutigen Überlappung

Es wird eine Kopie G_j von G_i erstellt und in G_i eine Kante der Länge [0, ∞] zwischen Knoten *A* und *B* eingefügt. In G_j wird eine Kante der Länge [0, ∞] zwischen Knoten *B* und *A* eingefügt. Dadurch sind nun zwei Graphen entstanden, die jeweils eine eindeutige Anordnung der Knoten *A* und *B* beinhalten. Man fährt nun mit dem in Abschnitt 8.1.2.1 beschriebenen Schritt des Verfahrens für G_i und G_j fort.

Aus dem Ereignisgraph in Abbildung 8-6 entstehen mittels dieses Verfahrens zwei eindeutige Ereignisgraphen. Ein Ereignisgraph besitzt zusätzlich eine Kante der Länge [0, 10] vom Präsentationsintervallendknoten von *VideoA* zum Präsentationsintervallanfangsknoten von *VideoB*. Der andere Ereignisgraph besitzt zusätzlich eine Kante der Länge [0, 10] vom Präsentationsintervallanfangsknoten von *VideoB* zum Präsentationsintervallendknoten von *VideoA*.

8.1.2.7 Überprüfung exklusiver Ressourcenanforderungen

Ein eindeutiger Ereignisgraph besteht aus einer Menge aufeinanderfolgender **Segmente**. Ein Segment beschreibt den Zeitraum zwischen zwei relevanten Knoten des Ereignisgraphs. Nach der Generierung eines eindeutigen Ereignisgraphs wird überprüft, ob sich die Präsentationsphase von Projektoren, welche dieselben Ressourcen exklusiv benötigen, in einem Segment überlappt. Ist dies der Fall, hat die mit diesem Graph assoziierte Präsentation einen unlösbaren Ressourcenkonflikt, und der Graph kann gelöscht werden.

Würden aus einer Spezifikation sehr viele eindeutige Ereignisgraphen entstehen, kann man, um Speicher zu sparen, auch nur eine gewisse Anzahl erzeugen, und nur wenn sich im Laufe der Präsentation der Bedarf ergibt, weitere eindeutige Ereignisgraphen erzeugen.

8.1.3 Attributierung eindeutiger Ereignisgraphen

Für die Ablaufplanung muß jeder Ereignisgraph G_i attributiert werden. Hierbei werden Variablen eingeführt, die flexible, aus einem eindeutigen Ereignisgraph ableitbare Eigenschaften repräsentieren, und zu einem Vektor H_i zusammengefaßt.

Jedem Knoten wird eine Variable x_j ($x_j \subseteq \Re$) zugeordnet, die sein Ereignisintervall beschreibt, und eine Variable t_j, die den aus dem Ereignisintervall x_j gewählten Ereigniszeitpunkt des Knotens beschreibt. Um die Vorbereitungsphase von Projektoren berücksichtigen zu können, wird ein weiteres Segment vor dem ersten Segment eingeführt. Eine Variable x_0 bzw. t_0 repräsentiert den Start dieses Segments.

Für jede Kante k des Ereignisgraphs wird eine Variable l_k ($l_k \subseteq \Re$) eingeführt, welche die Länge der Kante als sogenanntes **Längenintervall** beschreibt.

Für flexibel spezifizierte Präsentationsraten führt man wie folgt Variablen ein: Wurde für den Dienstgütebereich die Auswahlpolitik *first* oder *static* spezifiziert, wird genau eine Variable r^o ($r^o \subseteq \Re$) dem Startknoten S_K^o des entsprechenden Projektors o zugeordnet. Wurde die Auswahlpolitik *dynamic* spezifiziert, wird für jedes Segment s in dem der Projektor o Daten präsentiert eine separate Variable r_s^o ($r_s^o \subseteq \Re$) eingeführt.

Des weiteren wird für jedes Segment *s* pro betrachteter Ressource *R* eine Variable w_R^s ($w_R^s \in \Re^+$) eingeführt, die angibt, wieviel Einheiten der Ressource im Segment überschüssig sind. Ebenso wird für jedes Segment *s* pro betrachteter Ressource *R* eine Variable m_R^s ($m_R^s \in \Re^+$) eingeführt, die angibt, wieviel Einheiten der Ressource im Segment fehlen.

Für jeden Projektor *o*, dessen Mediendaten noch transferiert werden müssen, wird eine Variable b_o ($b_o \in \Re$) eingeführt, die beschreibt, wann mit dem Transfer der Mediendaten begonnen wird. Für jeden Projektor *o*, dessen Mediendaten noch dekodiert werden müssen, wird eine Variable c_o ($c_o \in \Re$) eingeführt, die beschreibt, wann mit der Dekodierung der Mediendaten begonnen wird.

Abbildung 8-7 zeigt den attributierten eindeutigen Ereignisgraph aus Abbildung 8-6. Aus jeder validen Belegung eines Vektors H_i bzw. der eingeführten Variablen läßt sich ein Ablaufplan ableiten.

8.2 Dynamische Ablaufplanung einer Dokumentpräsentation

Die Erstellung bzw. Anpassung von Ablaufplänen mittels des Tiempo-Ablaufplanungsverfahrens während der Präsentation eines Multimedia-Dokuments wird wie folgt durchgeführt:

1. **Bestimmung der ersten Ressourceninformationen**
   Mit Hilfe des Testbeds werden die Dekodierungskennzahlen für die Medientypen bestimmt (siehe Abschnitt 7.1). Dann werden die Ressourcenanforderungen anhand der Kennzahlen und der in der Dokumentspezifikation enthaltenen Informationen berechnet. Es werden der verfügbare Speicher und die verfügbaren CPU-Zyklen gemessen und die erste Prognose erstellt (siehe Abschnitt 7.2).

2. **Berechnung des initialen Ablaufplans**
   Der optimale Ablaufplan mit Speicher- und CPU-Einplanung wird entsprechend dem Algorithmus in Abschnitt 8.4 berechnet. Dieser Algorithmus bestimmt einen Ablaufplan iterativ, wobei in jedem Iterationsschritt eine Ressourceneinplanung, wie in Abschnitt 8.3 beschrieben, durchgeführt wird.

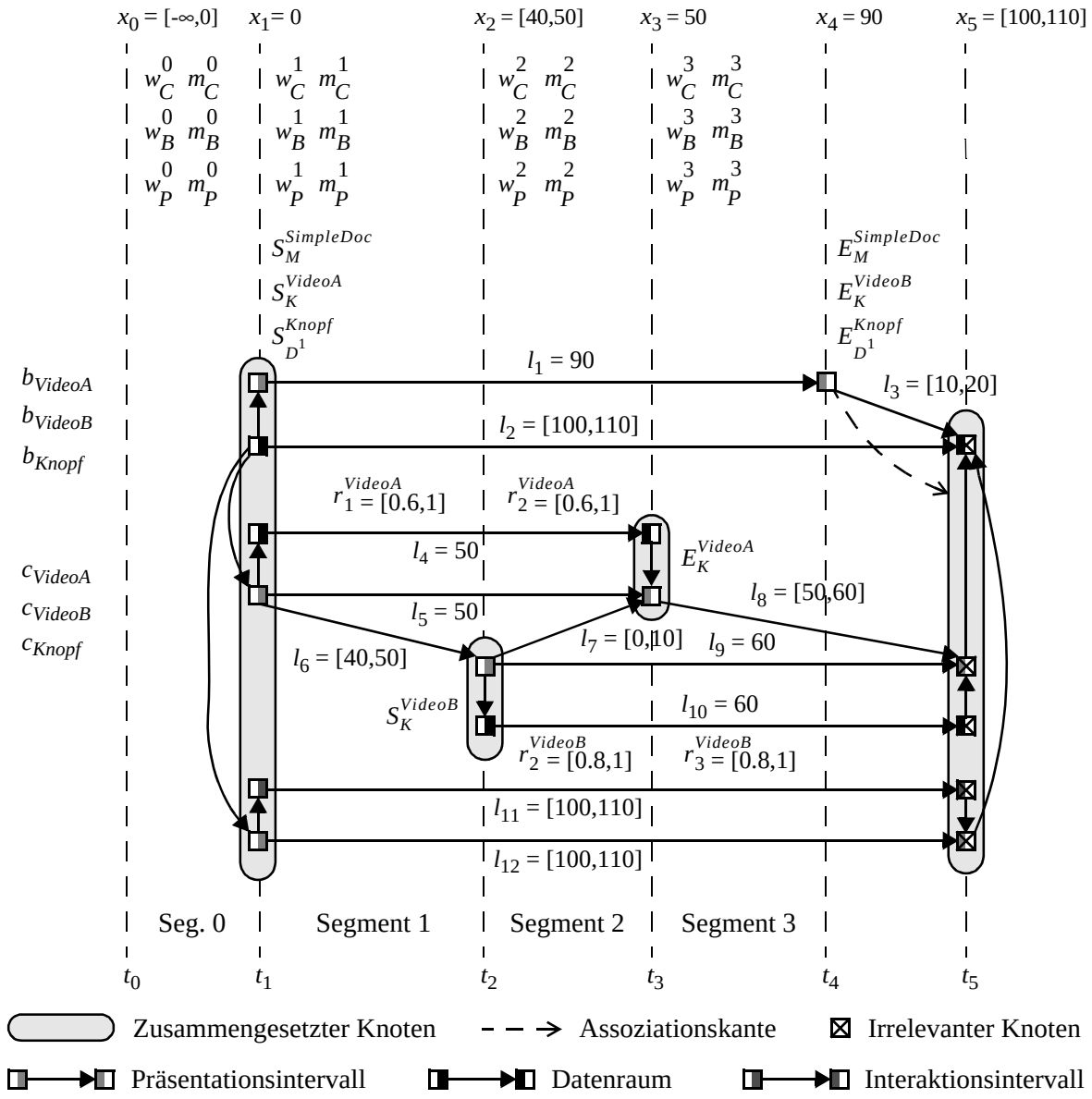


Abb. 8-7 : Attributierung eindeutiger Ereignisgraphen

3. Transfer der am frühesten präsentierten Mediendaten

Entsprechend des initialen Ablaufplans wird der Transfer der Medien initiiert, welche am frühesten präsentiert werden sollen.

4. Messung der verfügbaren Bandbreite

Während des Transfers der Medien wird die Bandbreite, die Kommunikationsverzögerung und der Jitter gemessen und eine erste Prognose für diese Werte berechnet (siehe Abschnitt 7.2).

5. **Adaption des Ablaufplans**

   Anhand der fortlaufenden Messung der verfügbaren Ressourcen Speicher, CPU-Zyklen, Bandbreite sowie der Kommunikationsverzögerung und des Jitters werden Adaptionen des Ablaufplans, wie in Abschnitt 8.5 beschrieben, berechnet, bis die Dokumentpräsentation beendet ist.

In den folgenden Abschnitten werden einzelne Schritte dieses Ablaufplanungsverfahrens näher erläutert.

8.3 Erstellung eines ressourcenoptimalen Ablaufplans

Ein eindeutiger Ereignisgraph kann durch Dienstgütebereiche implizierte Flexibilität enthalten. Der in diesem Abschnitt beschriebene heuristische Algorithmus ermöglicht es, aus einem eindeutigen Ereignisgraph einen Ablaufplan abzuleiten, der einen minimalen Ressourcenverbrauch impliziert.

8.3.1 Überprüfung der positionsabhängigen Ressourcenanforderungen

Der eindeutige Ereignisgraph wird durchlaufen. Für jedes Segment *s* wird ein Einplanungstest für positionsabhängige Ressourcenanforderungen durchgeführt, indem überprüft wird, ob die Ungleichungen (8-1) für das Ereignisintervall x_s erfüllt sind. Hierbei setzt man auf der rechten Seite der Ungleichungen die aktuellen Prognosefunktionen ein und auf der linken Seite die minimalen Präsentationsraten.

Sind für einige Segmente die Ungleichungen nicht erfüllt, sind zu wenig Ressourcen vorhanden. Ist dies der Fall, wird für die unerfüllten Ungleichungen in der entsprechenden Variablen m_R^s vermerkt wieviel Ressourcen fehlen und das Verfahren abgebrochen, da die durch den Ereignisgraph implizierte Präsentation zu viele Ressourcen benötigt.

Implizieren alle Graphen Ressourcenengpässe und die Präsentation soll dennoch fortgeführt werden, kann anhand der Werte der Variablen m_R^s entschieden werden, welcher Ereignisgraph längerfristig den geringsten Ressourcenengpaß verursacht.

8.3.2 Bestimmung ressourcenoptimaler Ereigniszeitpunkte

In diesem Schritt werden alle Indeterminismen aus dem Ereignisgraph eliminiert. Es wird für jeden Knoten ein Ereigniszeitpunkt t_j aus seinem Ereignisintervall x_j bestimmt, so daß eine möglichst große Menge freier Ressourcen zur Einplanung positionsunabhängiger Ressourcenanforderungen entsteht.

Der **ressourcenoptimale Ereigniszeitpunkt** für den Anfangsknoten eines Präsentationsintervalls, Interaktionsintervalls oder Datenraums ist der späteste in seinem Ereignisintervall, denn je später das Anfangsereignis eintritt, desto mehr Zeit ist vorhanden eine Präsentationsaktion oder einen Interaktionseffekt vorzubereiten. Der ressourcenoptimale Ereigniszeitpunkt für den Endknoten eines Präsentationsintervalls, Interaktionsintervalls oder Datenraums ist der früheste in seinem Ereignisintervall, denn je früher das Endereignis eintritt, desto eher können die allokierten Ressourcen zur Vorbereitung oder Präsentation anderer Medien eingesetzt werden. Ist ein Knoten sowohl Anfangsknoten als auch Endknoten, wird er wie ein Anfangsknoten behandelt, da dadurch der Beginn einer Präsentation so weit wie möglich hinausgezögert wird, wodurch die meisten Ressourcen zur Einplanung von Vorbereitungsaktionen verfügbar werden.

Zur Bestimmung der ressourcenoptimalen Ereigniszeitpunkte werden die Variablen x_j und l_k mit einem Werteintervall entsprechend der Längenmatrix initialisiert. Die Knoten des Graphen werden topologisch sortiert [AHU83] und in den folgenden Schritten immer in dieser Reihenfolge betrachtet.

Als erstes wird der Graph, entsprechend den sortierten Knoten, vorwärts durchlaufen. Für jeden Knoten j führt man hierbei die folgende Berechnung durch:

1. Das Ereignisintervall x_j des Knotens j wird mit Hilfe der Ereignisintervalle x_i seiner Vorgängerknoten i und der Längenintervalle l_k der Kanten k zu diesen Knoten entsprechend folgender Vorschrift berechnet:

$$x_j = x_j \cap (x_i + l_k), \qquad l_k = l_k \cap (x_j - x_i) \tag{8-2}$$

   Diese Berechnung ist notwendig, um die Änderung der Ereignisintervalle zu berücksichtigen, die sich bei der Anwendung von Schritt 2 auf die Vorgängerknoten ergeben hat. Da die Ereignisintervalle und Längenintervalle Werteintervalle sind, wird hierzu eine Inter-

vallalgebra verwendet. Das berechnete Ereignisintervall x_j umfaßt alle Zeitpunkte, an denen die mit dem Knoten assoziierten Ereignisse auftreten können. Im Vergleich zur Konsistenzprüfung ist hierbei dieses einfache Verfahren ausreichend, weil ein eindeutiger Ereignisgraph nur vollständig konsistente Werteintervalle enthält.

2. Ist der Knoten *j* ein relevanter Anfangsknoten eines Präsentationsintervalls oder Datenraums, wird der späteste Zeitpunkt in seinem Ereignisintervall x_j als Ereigniszeitpunkt t_j gewählt und x_j gleich diesem Zeitpunkt gesetzt.

Nun wird der Graph rückwärts, entsprechend den sortierten Knoten, durchlaufen. Für jeden Knoten *j* führt man hierbei die folgende Berechnung durch:

1. Das Ereignisintervall x_j des Knotens *j* wird mit Hilfe der Ereignisintervalle x_i seiner Nachfolgerknoten *i* und der Längenintervalle l_k der Kanten *k* zu diesen Knoten entsprechend folgender Vorschrift berechnet:

$$x_j = x_j \cap (x_i - l_k), \qquad l_k = l_k \cap (x_i - x_j) \tag{8-3}$$

   Dadurch werden Änderungen an Ereignisintervallen der Nachfolgerknoten berücksichtigt, die Schritt 2 impliziert.

2. Ist der Knoten *j* ein relevanter Endknoten eines Präsentationsintervalls oder Datenraums, wird der früheste Zeitpunkt in seinem Ereignisintervall x_j als Ereigniszeitpunkt t_j gewählt und x_j gleich diesem Zeitpunkt gesetzt.

In einem nächsten Durchlauf durch den Graph werden die Anfangs- und Endzeitpunkte von Interaktionsintervallen so bestimmt, daß die mit ihnen assoziierten Interaktionseffekte umsetzbar sind. Gibt es dann noch Wahlmöglichkeiten, wird für relevante Anfangsknoten von Interaktionsintervallen der späteste Zeitpunkt in ihrem Ereignisintervall als Ereigniszeitpunkt gewählt und für relevante Endknoten von Interaktionsintervallen der früheste Zeitpunkt in ihrem Ereignisintervall. Dadurch erhält man das kürzeste Interaktionsintervall.

Da irrelevante Knoten nur Platzhalter sind und die mit ihnen korrelierte Ressourcenanforderung durch einen entsprechenden relevanten Knoten eines umgebenden Multimedia-Projektors repräsentiert wird, werden sie zuletzt betrachtet. Enthält ein Graph irrelevante Knoten, wird er

ein weiteres Mal durchlaufen, hierbei wird zuerst das Ereignisintervall jedes irrelevanten Knotens korrigiert und dann ein beliebiger Ereigniszeitpunkt gewählt.

Abbildung 8-8 zeigt die mittels dieses Algorithmuses bestimmten Ereigniszeitpunkte für einen Ereignisgraph. Dieser Ereignisgraph wird einmal vorwärts und einmal rückwärts durchlaufen, dann sind alle Ereigniszeitpunkte bestimmt. In der Abbildung sind Werte in Ereignis- und Längenintervallen, die im Vorwärtsdurchlauf ausgeschlossen werden, einfach durchgestrichen und solche, die im Rückwärtsdurchlauf ausgeschlossen werden, zweifach durchgestrichen. Dementsprechend wird im Vorwärtsdurchlauf der Ereigniszeitpunkt des Präsentationsintervallanfangsknotens von *VideoB* und der *Audio*-Sequenz sowie des Datenraumendknotens des Dokuments *SimpleDoc* bestimmt. Der Präsentationsintervall- und Datenraumanfangsknoten von *VideoA* und *SimpleDoc* lagen schon vorab fest. Im Rückwärtsdurchlauf wird dann der Ereigniszeitpunkt des Präsentationsintervallendknotens von *VideoA* bestimmt.

In der unteren Hälfte von Abbildung 8-8 ist schematisch der Verbrauch an positionsabhängigen Ressourcen dargestellt, der sich durch die auf diese Weise bestimmten Ereigniszeitpunkte ergibt. Die dunklen Flächen repräsentieren die in den Präsentationsphasen benötigten Ressourcen. Die hellen Flächen zeigen, wo Ressourcen durch die Anwendung des Algorithmuses nicht in Präsentationsphasen verwendet werden und somit zusätzlich zur Erfüllung positionsunabhängiger Ressourcenanforderung in Vorbereitungsphasen zur Verfügung stehen.

8.3.3 Einplanung positionsunabhängiger Ressourcenanforderungen

Es wird versucht die positionsunabhängigen Ressourcenanforderungen einzuplanen. Indem die Einplanung vom Ende der Präsentation nach vorne durchgeführt wird, d.h. der Graph rückwärts durchlaufen wird, werden verfügbare Ressourcen möglichst gut genutzt.

Während der Einplanung wird in geordneten Listen D_C und D_B vermerkt, welche positionsunabhängigen Ressourcen einzuplanen sind. Die Liste D_C enthält Tupel der Form (o, V_r, C_r) mit der Bedeutung: Für die Vorbereitung des Projektors o sind V_r Bytes zu speichern und C_r CPU-Zyklen bereitzustellen. Die Liste D_C ist nach den Werten C_r aufsteigend geordnet. Wird ein Element der Liste betrachtet, ist dies immer das mit dem kleinsten C_r. Dadurch werden zuerst Projektoren mit geringem CPU-Bedarf in der Vorbereitungsphase eingeplant, wodurch die verfüg-

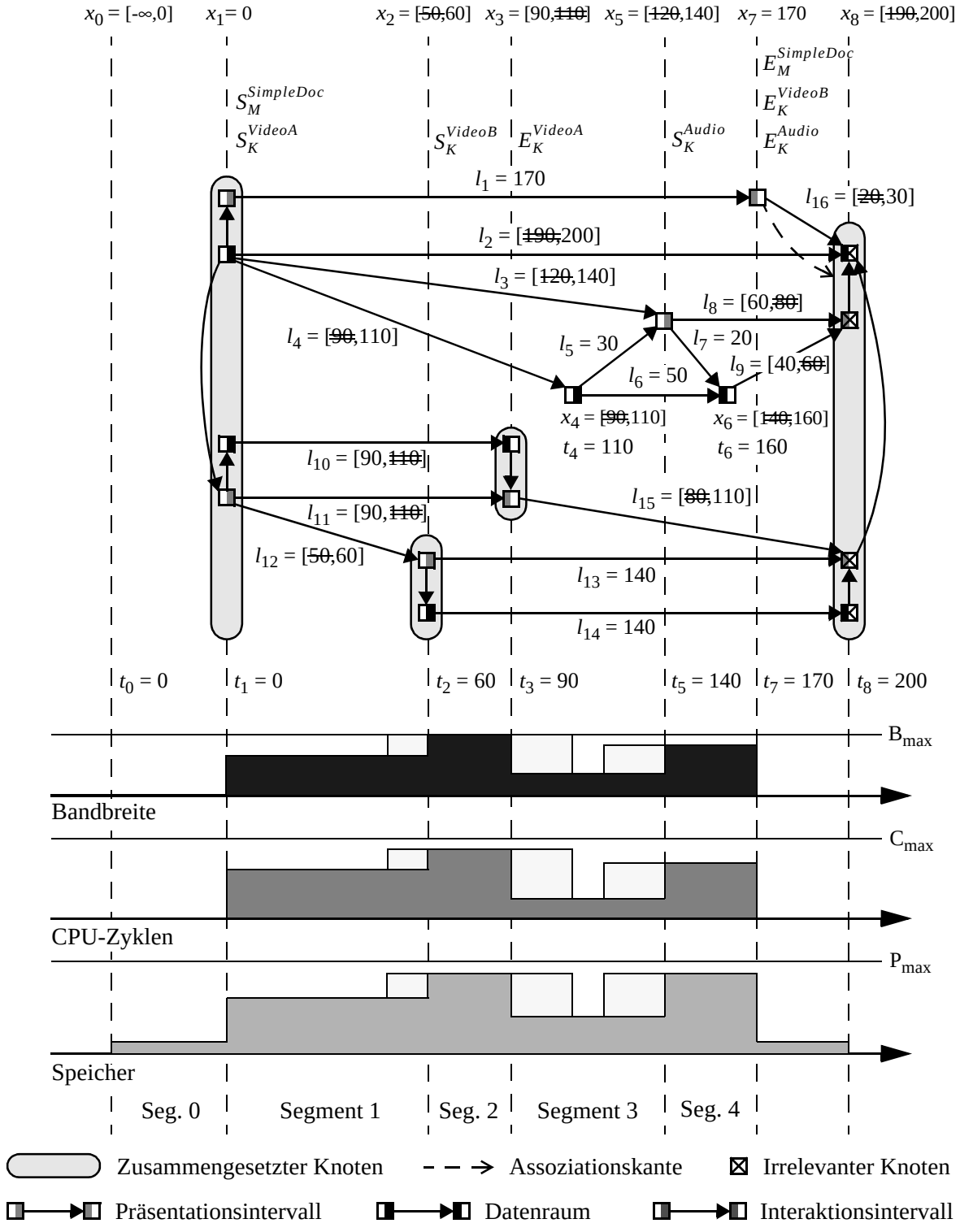


Abb. 8-8 : Bestimmung ressourcenoptimaler Ereigniszeitpunkte

bare Bandbreite optimal ausgenützt wird. Die Liste D_B enthält Tupel der Form (o, V_r, c_r) mit der Bedeutung: Für die Vorbereitung des Projektors o sind noch V_r Bytes zu transferieren und

der Transfer muß zum Zeitpunkt c_r abgeschlossen sein. Die Liste D_B ist nach den Zeitpunkten c_r absteigend geordnet. Wird ein Element der Liste betrachtet, ist dies immer das mit dem größten c_r. Auf diese Weise kann die verfügbare Bandbreite optimal genutzt werden.

In der folgenden algorithmischen Darstellung wird davon ausgegangen, daß die Knoten eines Ereignisgraphs so numeriert sind, daß ein Segment *s* beim Knoten mit der Nummer *s* beginnt und beim Knoten mit der Nummer *s*+1 endet.

Beim Durchlaufen des Graphs wird für jedes Segment *s*, das bei Ereigniszeitpunkt t_s beginnt und bei Ereigniszeitpunkt t_{s+1} endet, die folgende Berechnung durchgeführt:

1. Es wird die Menge der freien CPU-Zyklen w_C^s, der verfügbaren Bandbreite w_B^s und des freien Speichers w_P^s im Segment bestimmt. Diese entsprechen der Differenz zwischen rechter und linker Seite der ersten Ungleichung (8-1) zum Zeitpunkt t_{s+1}.

2. Ist der Knoten *s*+1 ein Startknoten $S_{D^1}^o$ diskreter Projektoren *o*, die in der Vorbereitungsphase CPU-Zyklen benötigen, werden für sie Tupel (o, V^o+P^o, C^o) in die Liste D_C eingefügt. Ist *s*+1 ein Startknoten $S_{D^1}^o$ diskreter Projektoren *o*, die in der Vorbereitungsphase nur Bandbreite benötigen, werden für sie Tupel (o, V^o, t_{s+1}) in die Liste D_B eingefügt. Ist der Knoten *s*+1 ein Startknoten S_M^o von Multimedia-Projektoren *o*, werden für sie Tupel (o, P^o, C^o) in die Liste D_C eingefügt. Ist *s*+1 ein Startknoten S_K^o kontinuierlicher Projektoren *o*, werden für sie Tupel (o, $2\,j^{y^o} r^o V^o$, t_{s+1}) in die Liste D_B eingefügt. r^o ist hierbei die Präsentationsrate des Projektors *o*, mit der er seine Präsentation beginnt. D_2-Phasen von Projektoren müssen nicht vorbereitet werden, da sie direkt auf eine kontinuierliche Präsentationsphase folgen und in ihnen die letzte Dateneinheit dieser Phase präsentiert wird.

3. Es wird geprüft, ob genug Puffer vorhanden ist, um die Daten der Projektoren in D_C und D_B, deren Präsentation in den Segmenten 0, ..., *s* vorbereitet werden muß, zu speichern. Falls die Ungleichung

$$\sum_{(o,\, V_r,\, C_r) \in\, D_C} V_r + \sum_{(o,\, V_r,\, c_r) \in\, D_B} V_r \le w_P^s \qquad (8\text{-}4)$$

nicht erfüllt ist, fehlt im Segment Speicher. Wieviel Speicher fehlt, vermerkt man in m_P^s.

4. Nun wird der CPU-Bedarf für Projektoren in D_C eingeplant. Es wird $\Delta t_C = t_{s+1} - t_s$ gesetzt und das erste Tupel (o, V_r, C_r) in D_C betrachtet:

   a. Falls $\Delta t_C w_C^s \geq C_r$ ist, wird $\Delta t_C = \Delta t_C - C_r / w_C^s$, $c^o = t_s + \Delta t_C$ gesetzt, das Tupel für o aus D_C entfernt und in D_B ein Tupel (o, V^o, c^o) eingefügt. Ist D_C leer, wird mit Schritt 5 fortgefahren, ansonsten wird das nächste Tupel in D_C betrachtet.

   b. Falls $\Delta t_C w_C^s < C_r$ ist, wird $C_r = C_r - \Delta t_C w_C^s$, $\Delta t_C = 0$ gesetzt und mit Schritt 5 fortgefahren.

5. Zur Einplanung des Bandbreitenbedarfs wird $\Delta t_B = t_{s+1} - t_s$ gesetzt und für das erste Tupel (o, V_r, c_r) in D_B die folgende Berechnung durchgeführt:

   a. Es wird $\Delta t_B = \min\{\Delta t_B, c_r - t_s\}$ gesetzt.

   b. Falls $\Delta t_B w_B^s \geq V_r$ ist, wird $\Delta t_B = \Delta t_B - V_r / w_B^s$, $b^o = t_s + \Delta t_B$ gesetzt und das Tupel aus D_B entfernt. Ist D_B leer, wird Schritt 1 auf Segment $s - 1$ angewendet, ansonsten wird das nächste Tupel in D_B betrachtet.

   c. Falls $\Delta t_B w_B^s < V_r$ ist, wird $V_r = V_r - \Delta t_B w_B^s$, $\Delta t_B = 0$ gesetzt und Schritt 1 auf Segment $s - 1$ angewendet.

Wurden alle Segmente, einschließlich Segment Null, auf diese Weise bearbeitet und es gibt auf einer der Listen noch Tupel, dann fehlen die entsprechenden Ressourcen für die zeitgerechte Vorbereitung dieser Projektoren. Bei der Bestimmung eines initialen Ablaufplans wird Segment Null nun soweit verbreitert, daß dieser Ressourcenbedarf erfüllt werden kann. Bei der Bestimmung einer Adaption während einer Dokumentpräsentation ist dies nicht möglich, d.h. die durch den eindeutigen Ereignisgraph implizierte Präsentation verursacht einen Ressourcenengpaß. Sind die Listen leer und die m_P^s-Werte gleich Null, kann ein Ablaufplan erstellt werden.

8.3.4 Ableitung des Ablaufplans

Ein Tiempo-Ablaufplan besteht aus einer Menge von Operationen der Form (t, a). Hierbei ist t der Zeitpunkt, zu dem die Aktion a auszuführen ist. Die Operationen sind nach aufsteigendem t geordnet. Aus einer Variablenbelegung kann wie folgt ein Ablaufplan erstellt werden.

Zur Vorbereitung der D_1-Präsentationsphase eines kontinuierlichen Projektors *o* wird eine Operation ($b_o - d_r^{y^o}$, *prepare_stream*(*o*, i_D^o, i_D^o, 1, 1)) eingeplant, welche die in dieser Phase auszuspielende Dateneinheit mit dem Medienzeitpunkt i_D^o vom Server y^o zum Präsentationsterminal transferiert. Diese Operation beschreibt, daß zum Zeitpunkt $b_o - d_r^{y^o}$ die Dateneinheiten mit den Medienzeitpunkten i_D^o bis i_D^o als kontinuierlicher Datenstrom mit der Geschwindigkeit 1 und der Datenrate 1 Dateneinheit pro Zeiteinheit transferiert werden sollen. Die Operation wird hierbei so eingeplant, daß die Antwortzeit $d_r^{y^o}$ des Servers, auf dem das mit dem Projektor assoziierte Medium gespeichert ist, berücksichtigt wird. Der Medienzeitpunkt i_D^o wird anhand der relativen Anordnung des Präsentationsintervalls und des Datenraums des Projektors im Ereignisgraph bestimmt.

Zur Vorbereitung der *K*-Präsentationsphase eines kontinuierlichen Projektors *o* wird die Operation ($b_o - d_r^{y^o}$, *prepare_stream*(*o*, i_K^o, $i_{s'+1}^o$, $\hat{v}_{s'}^o$, $\hat{r}_{s'}^o$)) eingeplant. Des weiteren werden für die Segmente *s* mit $b_o < t_s < c_o$ die Operationen ($t_s - d_r^{y^o}$, *prepare_stream*(*o*, i_s^o, i_{s+1}^o, $\hat{v}_s^o$, $\hat{r}_s^o$)) in den Ablaufplan eingefügt. Der Medienzeitpunkt i_K^o der ersten zu transferierenden Dateneinheit wird anhand der relativen Anordnung des Präsentationsintervalls und des Datenraums des Projektors im Ereignisgraph bestimmt. Ist *s*' das Segment in dem der Zeitpunkt b_o liegt, dann berechnet sich der Medienzeitpunkt der Dateneinheit mit welcher der Transfer im Segment $s' + 1$ beginnt als $i_{s'+1}^o = i_K^o + \hat{v}_{s'}^o(t_{s'+1} - b_o)$ und für alle anderen Segmente *s* mit $b_o < t_s < c_o$ als $i_s^o = i_{s-1}^o + \hat{v}_{s-1}^o(t_s - t_{s-1})$. Die Datenrate in einem Segment *s* berechnet sich als $\hat{r}_s^o = w_B^s / V^o$. Die Geschwindigkeit berechnet sich bei Projektoren mit dynamischer Präsentationsrate als $\hat{v}_s^o = (v^o / r_{S_K^o}^o)\,\hat{r}_s^o$ und bei Projektoren mit statischer Präsentationsrate als $\hat{v}_s^o = (v^o / r^o)\,\hat{r}_s^o$.

Zur Vorbereitung der D_1-Präsentationsphase eines diskreten Projektors *o* wird eine Operation ($b_o - d_r^{y^o}$, *prepare*(*o*)) eingeplant.

Für die Vorbereitung jeder D_1-Präsentationsphase wird zudem eine Operation (c_o, *decode*(*o*, i_D^o)) eingeplant. Für Multimedia-Projektoren plant man eine äquivalente Operation ein.

Für eine D_i-Präsentationsphase wird die Operation ($t_{S_{D^i}^o}$, *start_Di*(*o*)) zum Starten der Präsentation und ($t_{E_{D^i}^o}$, *stop*(*o*)) zum Beenden der Präsentation eingeplant. Für eine *K*-Präsentationsphase wird bei einem Projektor mit dynamischer Präsentationsrate für jedes Segment *s*,

$S_K^o \le s < E_K^o$ eine Transferoperation ($t_s - d_r^{y^o}$, *prepare_stream*(o, i_s^o, i_{s+1}^o, v^o, r_s^o)) eingeplant, wobei $i_s^o = i_{s-1}^o + v^o(t_s - t_{s-1})$ gilt. Die erste Transferoperation beginnt beim Medienzeitpunkt, bei dem die letzte Transferoperation in der Vorbereitungsphase endet. Die letzte Transferoperation endet beim Medienzeitpunkt $i_K^o + v^o(t_{E_K^o} - t_{S_K^o})$. Zudem wird für jedes Segment *s*, $S_K^o \le s < E_K^o$ eine Startoperation (t_s, *start_K*(o, i_s^o, v^o, r_s^o)) mit $i_s^o = i_{s-1}^o + v^o(t_s - t_{s-1})$ eingeplant. Die erste Startoperation beginnt mit Medienzeitpunkt $i_{S_K^o}^o = i_K^o$. Bei Projektoren mit statischer Präsentationsrate wird nur eine Transferoperation ($t_{S_K^o} - d_r^{y^o}$, *prepare_stream*(o, i_s^o, $i_K^o + v^o(t_{E_K^o} - t_{S_K^o})$, v^o, r^o)) eingeplant, die beim Medienzeitpunkt, bei dem die letzte Transferoperation in der Vorbereitungsphase endet, beginnt. Des weiteren wird nur eine Startoperation ($t_{S_K^o}$, *start_K*(o, i_K^o, v^o, r^o)) eingeplant. Schließlich plant man noch eine Stoppoperation ($t_{E_K^o}$, *stop*(o)) ein. Für Multimedia-Projektoren verfährt man analog.

Für Interaktionsintervalle werden entsprechende Start- und Stoppaktionen eingeplant, durch die der entsprechende Interaktionsmanager sensitiv bzw. insensitiv für Benutzerinteraktionen wird.

Wie die Operationen konkret umgesetzt werden, hängt von der Präsentationsumgebung ab. Beispielsweise könnten für Transferoperationen Ressourcen reserviert werden, wenn es die Präsentationsumgebung zuläßt [ABF97, TaHo94, NaSm95] oder es könnte beim Ausspielen von kontinuierlichen Medien ein entsprechendes CPU-Schedulingverfahren, wie beispielsweise [Bart94], angewendet werden.

8.4 Berechnung eines optimalen Ablaufplans

Bei der Suche nach einem optimalen Ablaufplan werden die eindeutigen Ereignisgraphen in der Reihenfolge absteigender Dienstgüte betrachtet, welche sie durch die enthaltenen Präsentationsalternativen implizieren. Man beginnt mit den Ereignisgraphen, welche die beste Dienstgüte implizieren. Dies können mehrere sein, da alle aus einer Kombination von Präsentationsalternativen entstandenen Ereignisgraphen die gleiche Dienstgüte auf Objektebene implizieren. Es wird nun versucht mit dem Verfahren in Abschnitt 8.3 eine ressourcenoptimale Variablenbelegung zu bestimmen. Gibt es Ereignisgraphen für die dies ohne Ressourcenkonflikt gelingt, geht man zur Ablaufplanbestimmung über. Verursachen alle diese Ereignisgraphen einen Ressourcenengpaß, werden die nächstschlechteren Ereignisgraphen betrachtet usw. Gibt es keinen Ereignisgraph ohne Ressourcenkonflikt und soll die Präsentation des Dokuments dennoch statt-

finden bzw. fortgeführt werden, wird der Ereignisgraph gewählt, der den geringsten Ressourcenengpaß verursacht, indem die durch das Verfahren in Abschnitt 8.3 berechneten m_R^s-Werte betrachtet werden.

Eine ressourcenoptimale Variablenbelegung bzw. der daraus erzeugte Ablaufplan ist im allgemeinen nicht optimal im Hinblick auf die gesetzten Prioritäten in Dienstgütebereichen, da diese vom Algorithmus in Abschnitt 8.3 nicht berücksichtigt werden. Prioritäten in Dienstgütebereichen können auf unterschiedliche Weise bei der Bestimmung einer Variablenbelegung berücksichtigt werden. Im folgenden werden zwei unterschiedliche Ansätze vorgestellt.

8.4.1 Optimierung der Summe von Prioritäten

Die Prioritätsfunktion, welche ein Dienstgütebereich $[x_1{:}p_1, x_2{:}p_2, ..., x_k{:}p_k, x_{k+1}{:}p_{k+1}]$ über den durch sich definierten Wertebereich $x_1 \le x \le x_{k+1}$ beschreibt, lautet:

$$W(x) = \begin{cases} p_1 + c_1(x - x_1), & x_1 \le x < x_2 \\ p_2 + c_2(x - x_2), & x_2 \le x < x_3 \\ \dots & \\ p_k + c_k(x - x_k), & x_k \le x \le x_{k+1} \end{cases} \qquad (8\text{-}5)$$

$$c_i = \frac{p_{(i+1)} - p_i}{x_{(i+1)} - x_i}, \quad i = 1, \dots, k$$

Enthält ein attributierter Ereignisgraph $n = q + p$ Variablen, von denen q Längenintervalle $l_1, \dots, l_q$ und p Präsentationsraten sind, wobei die unterschiedlichen Raten eines Projektors i $(i = 1, \dots, p)$ in den Segmenten $s_1^i, \dots, s_k^i$ durch eine Variablenmenge $\{ r_{s_1^i}^i, \dots, r_{s_k^i}^i \}$ repräsentiert werden. Sind zudem die Prioritätsfunktionen W_l^i, $i = 1, \dots, q$ und W_r^i, $i = 1, \dots, p$ durch Dienstgütebereiche definiert, dann ist eine Variablenbelegung **optimal**, wenn sie eine Teilbelegung $x = (l_1, \dots, l_q, r_{s_1^1}^1, \dots, r_{s_k^1}^1, r_{s_1^2}^2, \dots, r_{s_k^2}^2, \dots, r_{s_1^p}^p, \dots, r_{s_k^p}^p)^T$ enthält, so daß die Funktion

$$Z(x) = \sum_{i=1}^{q} W_l^i(l_i) + \sum_{i=1}^{p} \left[\frac{1}{s_k^i - s_1^i + 1} \sum_{j=s_1^i}^{s_k^i} W_r^i(r_j^i) \right] \to \text{Max.} \qquad (8\text{-}6)$$

ihr Maximum erreicht, d.h. die Summe der Prioritäten der gewählten Werte aus den Dienstgütebereichen maximal ist. Für Präsentationsraten wird der Durchschnitt über alle Segmente

gebildet, in denen ein Medium präsentiert wird. Somit ist das Finden einer optimalen Variablenbelegung ein Optimierungsproblem.

Die Funktionen (8-5) müssen nicht konvex sein, wodurch man durch Anwendung von Optimierungsverfahren für konvexe Probleme lediglich eine suboptimale Lösung findet, die nicht die optimale Lösung sein muß [Mino86, Neum75]. Das liegt daran, daß diese Optimierungsverfahren ausgehend von einer gültigen Lösung des Mathematischen Programms iterativ eine immer bessere Lösung anhand der Ableitung der konvexen Zielfunktion bestimmen, bis der Zielfunktionswert nicht mehr verbessert werden kann, ohne den Lösungsraum zu verlassen. Ist die Zielfunktion nicht konvex, hat sie jedoch mehrere Suboptima im Lösungsraum und man bleibt im ersten gefundenen Suboptimum hängen. Im folgenden wird deshalb ein Optimierungsverfahren beschrieben, das auf einem Greedy-Ansatz basiert und auch für spezielle nicht-konvexe Optimierungsprobleme geeignet ist.

Dieses Optimierungsverfahren ist auf Optimierungsprobleme der Form (8-7) mit einer linearen Zielfunktion $Z(x)$ und beliebigen konvexen oder nicht-konvexen Nebenbedingungen $g_i(x) \le 0$ anwendbar:

$$\begin{aligned} Z(x) &= \sum_{j=1}^{n} c_j x_j \rightarrow \text{Max.} \\ g_i(x) &\le 0, \qquad i = 1, \ldots, m \\ x_j &\ge 0, \qquad j = 1, \ldots, n \\ c_j &\in \Re, \qquad j = 1, \ldots, n \end{aligned} \tag{8-7}$$

Das Optimierungsverfahren findet einen Vektor $x \in \Re^n$, der die Nebenbedingungen $g_i(x) \le 0$, $x_j \ge 0$ erfüllt und zugleich einen maximalen Wert für $Z(x)$ liefert. Ein solcher Vektor wird optimale Lösung x^* von (8-7) genannt.

Obwohl sich viele Optimierungsproblem in die Form (8-7) bringen lassen, besteht eine Einschränkung für die Anwendung des Optimierungsverfahrens: Der Minimalwert und der Maximalwert jeder Variablen x_j muß anhand der Nebenbedingungen $g_i(x) \le 0$ bestimmt werden können. Beim optimalen Variablenbelegungsproblem im Rahmen der Ablaufplanung ist dies der Fall.

Die optimale Lösung von (8-7) kann wie folgt gefunden werden:

1. Für jedes Variablenpaar x_i, x_j wird ein Wert f_{ij} bestimmt, der beschreibt, ob die Variablen **unabhängig** sind $(f_{ij} = 0)$ oder nicht $(f_{ij} = 1)$. Zwei Variablen sind unabhängig, wenn die Vergrößerung des Wertes der einen Variablen nicht dazu führt, daß aufgrund einer Nebenbedingung der Wert der anderen Variablen verkleinert werden muß. Sind Variablen unabhängig voneinander, kann ihr Wert gleichzeitig erhöht oder verringert werden.

2. Es wird eine Menge $I_a = \{1, \ldots, n\}$ definiert, welche die Indizes der Variablen enthält, deren Werte noch nicht festliegen, und eine Menge $I_p = \varnothing$, welche die Indizes der Variablen enthält, deren optimaler Wert bereits bestimmt ist. Dann bestimmt man die optimale Lösung von (8-7) wie folgt:

   a. Anhand der Nebenbedingungen $g_i(x) \leq 0, x_j \geq 0$ wird für jede Variable x_j der minimale $(\underline{x}_j)$ und maximale $(\bar{x}_j)$ Wert berechnet, den sie annehmen kann. Der Index jeder Variablen, für die gilt $\underline{x}_j = \bar{x}_j, j = 1, \ldots, n$, wird aus der Menge I_a entfernt und zur Menge I_p hinzugefügt. Ist die Menge I_a leer ist die optimale Lösung bestimmt.

   b. Man bestimmt das Tripel

$$
\begin{aligned}
(z, \Delta x, I_z) \quad & \text{mit} \quad |z| \rightarrow \text{Max.}, z = \Delta x \sum_{i \in I_z} c_i, \Delta x = \min_{p \in I_z}\{\bar{x}_p - \underline{x}_p\}, I_z \subseteq I_a \\
& \text{und} \quad \forall i \in I_z\colon c_i \neq 0 \quad \text{falls} \quad |I_z| > 1 \\
& \text{und} \quad \forall (i, j) \in I_z \times I_z\colon f_{ij} = 0
\end{aligned}
\tag{8-8}
$$

   Die Tripel haben folgende Semantik: Wird der Wert jeder Variablen $x_j, j \in I_z$ um Δx vergrößert, erhöht sich der Wert der Zielfunktion $Z(x)$ um z. Δx ist der maximale Wert, um den alle Variablen in I_z verändert werden können, ohne eine der Nebenbedingungen zu verletzen. $Z(x)$ wird am meisten vergrößert, wenn das Tripel mit dem größten z gewählt wird. Gibt es mehrere Tripel, die denselben z-Wert haben, wird eins davon ausgewählt. Die zusätzlichen Bedingungen fordern, daß die Indexmengen I_z nur Indizes von Variablen enthalten, die unabhängig voneinander sind und deren Koeffizient in $Z(x)$ ungleich Null ist, falls I_z mehrere Indizes enthält. Dadurch wird sichergestellt, daß nur valide Indexmengen entstehen und diese möglichst klein sind.

c. Ist $z > 0$, wird $\underline{x}_j = \underline{x}_j + \Delta x,\ j \in I_z$ gesetzt, sonst wird $\bar{x}_j = \bar{x}_j - \Delta x,\ j \in I_z$ gesetzt. Auf diese Weise wird der Wertbereich der Variablen in I_z so eingeschränkt, daß die entsprechende Erhöhung der Zielfunktion um z gegeben ist. Danach fährt man mit Schritt 2a fort.

Der folgende Widerspruchsbeweis zeigt, daß der Algorithmus die optimale Lösung findet. Denn wäre dies nicht der Fall, muß eine der folgenden Möglichkeiten bestehen:

1. Es gibt mindestens eine Variable x_j, deren Werte je nach Vorzeichen von c_j erhöht bzw. erniedrigt werden kann, um $Z(x)$ zu erhöhen. Dies steht jedoch im Widerspruch zu Schritt 2b und 2c, da für eine Variable x_j je nach Vorzeichen von c_j immer der größte bzw. kleinste Wert innerhalb des durch die Nebenbedingungen von (8-7) aufgespannten Lösungsraums gewählt wird.

2. Indem die Werte einer Menge Variablen I_1 um Δx_1 erniedrigt werden, können die Werte einer Menge Variablen I_2 um Δx_2 erhöht werden, wodurch der Wert von $Z(x)$ steigt. D.h. es muß gelten $\sum_{j \in I_1} c_j \Delta x_1 < \sum_{j \in I_2} c_j \Delta x_2$. Dies steht jedoch im Wiederspruch zu Schritt 2b, da das Tripel für die Indexmenge I_2 vor I_1 gewählt werden würde und die Variablen in I_2 in Schritt 2c je nach Vorzeichen von z auf den größten bzw. kleinsten Wert gesetzt werden würden, so daß eine weitere Erhöhung nicht mehr möglich ist, ohne den Lösungsraum zu verlassen.

Ist die Zielfunktion (8-6) linear, kann der Algorithmus wie folgt auf einen eindeutigen Ereignisgraph angewendet werden.

In einem eindeutigen Ereignisgraph haben alle relevanten Knoten eine eindeutige Anordnung. Lediglich der Abstand der relevanten Knoten kann variabel sein. Man kann deshalb zwischen zwei benachbarten Knoten je eine sogenannte **Distanzkante** h einführen und deren Länge l_h durch eine Variable beschreiben. Verlängert man eine Distanzkante h um eine Einheit, verlängern sich auch die Kanten, die parallel zu h liegen, um eine Einheit, wodurch sich der Wert der Zielfunktion (8-6) um den Wert $z = \sum_{i \in Q_h} c_i$ erhöht. Hierbei ist Q_h die Menge der Kanten, die parallel zu h liegen. Dementsprechend kann man für alle Distanzkanten $c_h = \sum_{i \in Q_h} c_i$ und für alle anderen Kanten $c_i = 0$ setzen, wodurch sich die Struktur des Optimierungsproblems vereinfacht. Für Präsentationsraten ist keine solche Vereinfachung möglich.

Die f_{ij}-Werte für Distanzkantenpaare lassen sich wie folgt bestimmen: Die Länge zweier Distanzkanten *i* und *j* ist unabhängig, wenn $Q_i \cap Q_j = \varnothing$ gilt. Denn dann gibt es keine Kante, welche die Länge des Pfades einschränkt, auf dem beide Distanzkanten liegen, wodurch man beide Kanten ohne direkte gegenseitige Beeinflussung verlängern oder verkürzen kann. Präsentationsraten sind unabhängig von Distanzkantenlängen, somit sind die entsprechenden f_{ij}-Werte gleich Null.

Die f_{ij}-Werte für Ratenvariablenpaare lassen sich wie folgt bestimmen: Es sei Q_r die Menge der Präsentationsratenvariablen, die in denselben Segmenten wie die Präsentationsratenvariable *r* liegt. Zwei Ratenvariablen *i* und *j* sind unabhängig, wenn $Q_i \cap Q_j = \varnothing$ gilt. Denn dann bedingt eine simultane Erhöhung von *i* und *j* einen Anstieg des Ressourcenverbrauchs in unterschiedlichen Segmenten und die Variablen beeinflussen sich nicht direkt.

Um in Schritt 2b das optimale Tripel zu finden, verfährt man wie folgt: Das optimale Tripel $(z, \Delta x, I_z)$ wird anhand der Werteintervalle der Variablen l_i und r_s^i bestimmt. Da in diesen Werteintervallen die positionsunabhängigen Ressourcenanforderungen nicht berücksichtigt sind, kann hierbei nur ein vorläufiger Wert $\Delta\tilde{x}$ bestimmt werden, der mittels des Algorithmus aus Abschnitt 8.3 korrigiert werden muß. Hierzu führt man Schritt 2c für $\Delta\tilde{x}$ aus und wendet den Algorithmus aus Abschnitt 8.3 an. Gibt es einen Ressourcenengpaß, wird das Intervallhalbierungsverfahren auf $\Delta\tilde{x}$ angewendet, bis das größte Δx gefunden ist, das keinen Ressourcenengpaß verursacht. Da das so ermittelte Δx nun aber nicht mehr das größte *z* implizieren muß, werden alle Tripel bestimmt, deren *z*-Wert, berechnet mit Hilfe eines vorläufigen $\Delta\tilde{x}$-Werts, größer als der *z*-Wert des besten Tripels ist. Für diese Tripel wird ebenfalls das größte Δx mittels des Intervallhalbierungsverfahrens bestimmt. Aus all diesen Tripeln wird schließlich das mit dem größten *z*-Wert als optimales Tripel gewählt.

Ist die Zielfunktion (8-6) nicht linear, geht man wie folgt vor: Für jeden Dienstgütebereich wird nur ein Teilintervall betrachtet, d.h. der Wertebereich zwischen zwei Stützstellen. Dadurch ergibt sich wiederum eine lineare Zielfunktion (8-6), auf die der oben beschriebene Algorithmus angewendet wird. Um die optimale Lösung möglichst schnell zu finden, werden Teilintervall-Kombinationen in einer bestimmten Reihenfolge betrachtet. Abbildung 8-9 zeigt die durch zwei Dienstgütebereiche definierte Prioritätsfunktionen. Der linke Dienstgütebereich besteht aus zwei Teilintervallen T_{11}, T_{12}, der rechte Dienstgütebereich aus drei Teilintervallen T_{21}, T_{22},

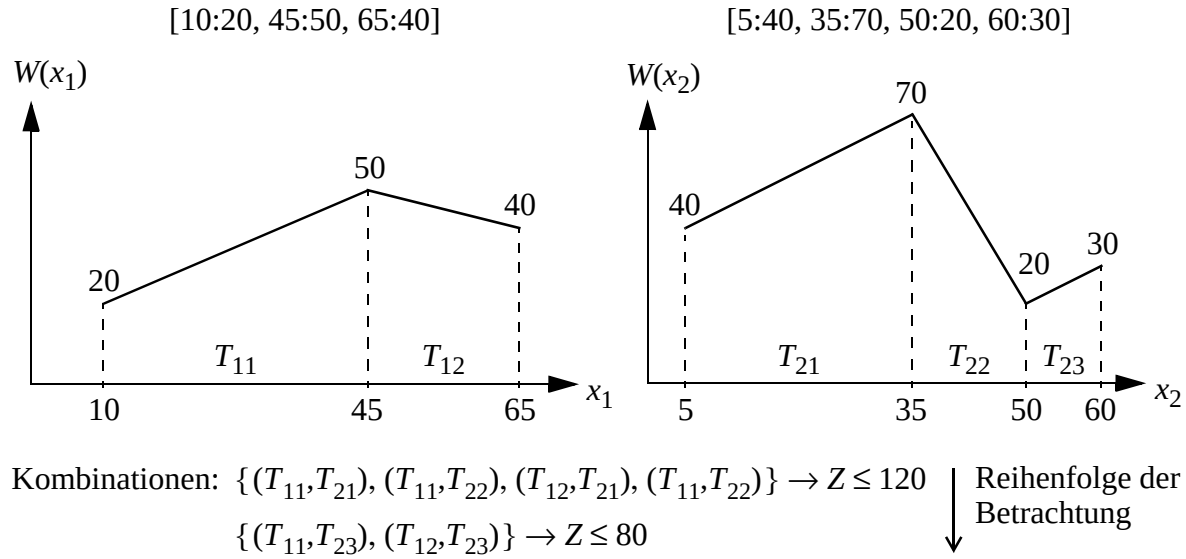


Abb. 8-9 : Kombination von Teilintervallen aus Dienstgütebereichen

T_{23}. Betrachtet man die Kombinationen der Teilintervalle, stellt man fest, daß es zwei Kombinationen gibt, durch die der Zielfunktionswert höchstens 80 sein kann und daß es vier Kombinationen gibt, durch die der Zielfunktionswert höchstens 120 sein kann. Werden diese vier Kombinationen zuerst betrachtet und es ergibt sich für die beste Kombination ein Zielwert über 80, müssen die anderen Kombinationen nicht mehr betrachtet werden, da keine bessere Lösung gefunden werden kann.

Verallgemeinert bedeutet dies, betrachtet man die Teilintervallkombinationen entsprechend den absteigenden maximalen Zielfunktionswerten, die sie implizieren, müssen die Kombinationen, die einen maximalen Zielfunktionswert unterhalb der besten gefundenen Lösung implizieren, nicht betrachtet werden.

Problematisch an diesem Ansatz ist, daß er NP-vollständig und damit sehr zeitintensiv ist.

8.4.2 Prioritäten steuern Reihenfolge der Wertzuweisung

Ein anderer Ansatz Prioritäten aus Dienstgütebereichen zu berücksichtigen ist, die Reihenfolge der Zuweisung von Werten für flexible Attribute durch die Prioritäten zu steuern.

Hierbei sucht man aus allen Dienstgütebereichen den Teil eines Teilintervalls heraus, der die größte Priorität impliziert und in dem die Prioritätsfunktion die geringste Steigung hat. Dann

bestimmt man für das entsprechende Attribut mit Hilfe des Intervallhalbierungsverfahrens und dem Algorithmus aus Abschnitt 8.3 den Wert mit der größten Priorität in diesem Intervall. Danach sucht man das nächste Teilintervall mit der größten Priorität usw., bis für alle Attribute, für die ein Dienstgütebereich mit Prioritäten spezifiziert wurde, ein Wert gefunden ist.

Würde dieses Verfahren auf die Dienstgütebereiche in Abbildung 8-10 angewendet, würde man im ersten Schritt das Teilintervall T_1 aus dem rechten Dienstgütebereich wählen. Ist für das entsprechende Attribut ein Wert innerhalb dieses Teilintervalls gefunden ist, würde man das Teilintervall T_2 aus dem linken Dienstgütebereich wählen und für dieses Attribut einen Wert bestimmen.

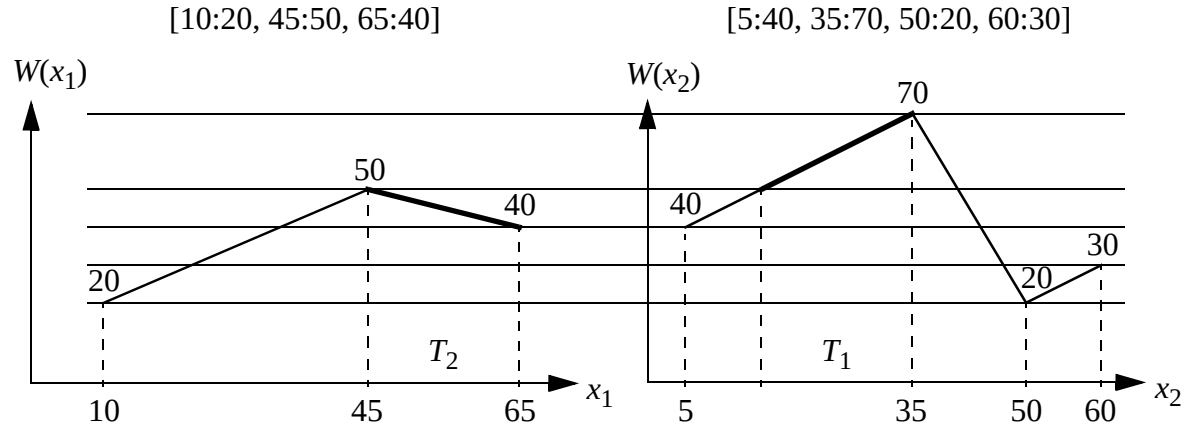


Abb. 8-10 : Auswahl von Teilintervallen aus Dienstgütebereichen

Der Vorteil dieses Optimierungsansatzes ist, daß bei jeder Iteration ein Attribut einen Wert zugewiesen bekommt, d.h. spätestens nach *n* Iterationen ist eine optimale Variablenbelegung bestimmt, falls es *n* flexible Attribute gibt. Das Verfahren hat somit eine polynominale Komplexität.

8.5 Adaption von Ablaufplänen

Ablaufpläne werden angepaßt, wenn sich die verfügbaren Ressourcen ändern oder eine Benutzerinteraktion auftritt.

8.5.1 Adaption bei Ressourcenänderungen

Ändert sich einer der gemessenen Ressourcenwerte um ein vorgegebenes Delta, wird eine Adaption des Ablaufplans berechnet. Hierbei wird als erstes, auf der Basis der Meßergebnisse der aktuellen positionsabhängigen Ressourcenanforderungen, der noch ausstehende Ressourcenbedarf hinsichtlich des zu transferierenden Datenvolumens und der CPU-Zyklen in bereits begonnen Vorbereitungsphasen bestimmt, sowie die Prognosefunktionen aktualisiert.

Indem die Ereignisintervalle und Längenintervalle in Ereignisgraphen entsprechend dem Verlauf der Präsentation laufend aktualisiert werden, kann nun mittels des Verfahrens in Abschnitt 8.4 ein neuer Ablaufplan bestimmt werden, der die veränderte Ressourcensituation berücksichtigt. Änderungen, die den schnelleren Transfer von Dateneinheiten oder den Transfer neuer Dateneinheiten betreffen, werden aufgrund der Kommunikationsverzögerung zu den Medien-Servern nicht sofort wirksam. Dementsprechend müssen Transferaktionsänderungen um die entsprechende Kommunikationsverzögerung verzögert eingeplant werden. Im Gegensatz dazu können Veränderungen hinsichtlich des CPU-Bedarfs und des Speicherbedarfs sofort umgesetzt werden, und somit auch ohne Verzögerung eingeplant werden.

Adaptionen aufgrund von Änderungen in der Kommunikationsverzögerung können unsichtbar für den Nutzer ablaufen. Da beispielsweise eine Verlängerung der Kommunikationsverzögerung zu einem Server nur dazu führen kann, daß das Senden einer Nachricht früher eingeplant wird.

Im Rahmen von Tests haben sich die in Tabelle 8-1 dargestellten Deltas für die Änderung von verfügbaren Ressourcen als sinnvoll erwiesen.

CPU	Speicher	Bandbreite	Antwortzeit	Jitter
5 %	100 KB	3 KB/s	1 s	0.5 s

Tabelle 8-1: Tolerierte Ressourcenänderungen

8.5.2 Adaption bei Interaktionen

Tritt eine Interaktion auf, wird mit der Vorbereitung der Interaktionseffekte begonnen. Dazu müssen die Vorbereitungsaktionen unter Berücksichtigung der Ressourcensituation in den

Ablaufplan eingefügt werden. Würde die Ressourcensituation nicht berücksichtigt werden, könnte es passieren, daß die Vorbereitungsaktionen aufgrund fehlender Ressourcen niemals abgeschlossen werden.

Da das Ausführen dieser Aktionen Zeit benötigt, verzögert sich das Herstellen des nächsten Präsentationszustands. In einer orchestrierten Dokumentpräsentation muß diese Verzögerung überbrückt werden. Die Präsentation anzuhalten, ist hierbei keine geeignete Reaktion. Es ist besser, die Präsentation fortzuführen, so als ob die Interaktion nicht aufgetreten wäre, bis es dann möglich ist, den entsprechenden Präsentationszustand herzustellen. Während der Wartephase werden jedoch keine weiteren Interaktionen akzeptiert und es werden auch keine Adaptionen durchgeführt, die das Umsetzen der Interaktionseffekte unmöglich machen, wie beispielsweise das vorzeitige Beenden eines Videos dessen Geschwindigkeit verändert werden soll. Durch die Wartephase kann es passieren, daß die Präsentation der beeinflußten Projektoren laut Ablaufplan beendet werden muß, ehe die Vorbereitungsaktionen zur Umsetzung der Interaktionseffekte abgeschlossen sind. Tritt dieser Fall ein, wird der Interaktionseffekt nicht umgesetzt.

8.6 Präsentationssteuerung

Die Präsentation eines Dokuments wird gemäß dem Ablaufplan gesteuert. Dazu wird zum berechneten Zeitpunkt die jeweilige Operation auf dem entsprechenden Dokumentelement ausgeführt.

8.6.1 Dokumentation der Präsentation

Um beim Auftreten von Interaktionen den Ablaufplan geeignet anpassen zu können, werden Informationen über den bisherigen Ablauf der Präsentation benötigt. Um beispielsweise eine Benutzerinteraktion *jumpRelative(-10)*, d.h. einen Rücksprung um zehn Sekunden durchführen zu können, muß bekannt sein, in welchem Zustand sich die Präsentation zum entsprechenden Zeitpunkt befunden hat. Denn nur dann kann dieser Zustand wiederhergestellt werden und die Präsentation entsprechend fortgeführt werden.

Eine Präsentation wird dokumentiert, indem bei jedem Ausführen einer Operation des Ablaufplans die Operation in einen Operationsspeicher eingetragen wird. Der Operationsspeicher ent-

hält somit alle durchgeführten Operationen, wodurch jeder vergangene Präsentationszustand wiederhergestellt werden kann.

8.6.2 Präsentationskontrolle von Dokumentelementen

Die Präsentation von Dokumentelementen wird mittels Zustandsgraphen kontrolliert. Wird entsprechend dem Ablaufplan eine Operation auf dem Dokumentelement ausgeführt oder tritt eine Interaktion auf, geht das Dokumentelement in den entsprechenden Zustand über. Für jedes Dokumentelement wird ein separater typabhängiger Zustandsgraph verwaltet. Die Zustandsgraphen für Intervalloperatoren, Auswahlgruppen, Interaktionsmanager und Reaktionsoperatoren sind identisch zu den in Abbildung 3-6 dargestellten Zustandsgraphen. Der Zustandsgraph für Projektoren ist, wie in Abbildung 8-11 zu sehen ist, erweitert, um Vorbereitungsphasen berücksichtigen zu können. Jeder Zustandsübergang zwischen (Haupt-)Zuständen, der es erfordert, daß bestimmte Mediendaten vorhanden sind, führt über zwei Zwischenzustände:

- ***prepare a***: In diesen Zustand geht der Projektor über, wenn die Vorbereitung der Mediendaten für den nächsten Hauptzustand beginnt und der Zustandswechsel durch die Aktion *a* angestoßen wurde.

- ***ready a***: In diesen Zustand geht der Projektor über, wenn die Vorbereitung der Mediendaten für den nächsten Hauptzustand abgeschlossen ist und der Zustandswechsel durch die Aktion *a* angestoßen wurde.

Mit Hilfe dieser Zwischenzustände verläuft die Präsentation eines Mediums wie folgt: Am Anfang befindet sich der Projektor im Hauptzustand *active*, d.h. er nimmt an der Präsentation teil. Wird entsprechend des Ablaufplans mit der Vorbereitungsphase begonnen, geht der Projektor in den Zwischenzustand *prepare start* über. Ist die Vorbereitungsphase des Projektors abgeschlossen, geht er in den Zwischenzustand *ready start* über. Beginnt laut Ablaufplan die Präsentationsphase, geht er in den Hauptzustand *running* über. Endet die Präsentationsphase geht er wieder in den Hauptzustand *active* über.

Für jede Interaktionsart gibt es ebenfalls zwei Zwischenzustände. Während der Vorbereitung von Interaktionseffekten befindet sich der Projektor solange in dem jeweiligen Zwischenzustand, bis die Dateneinheiten zur Verfügung stehen. Während dieser Phase wird die Präsenta-

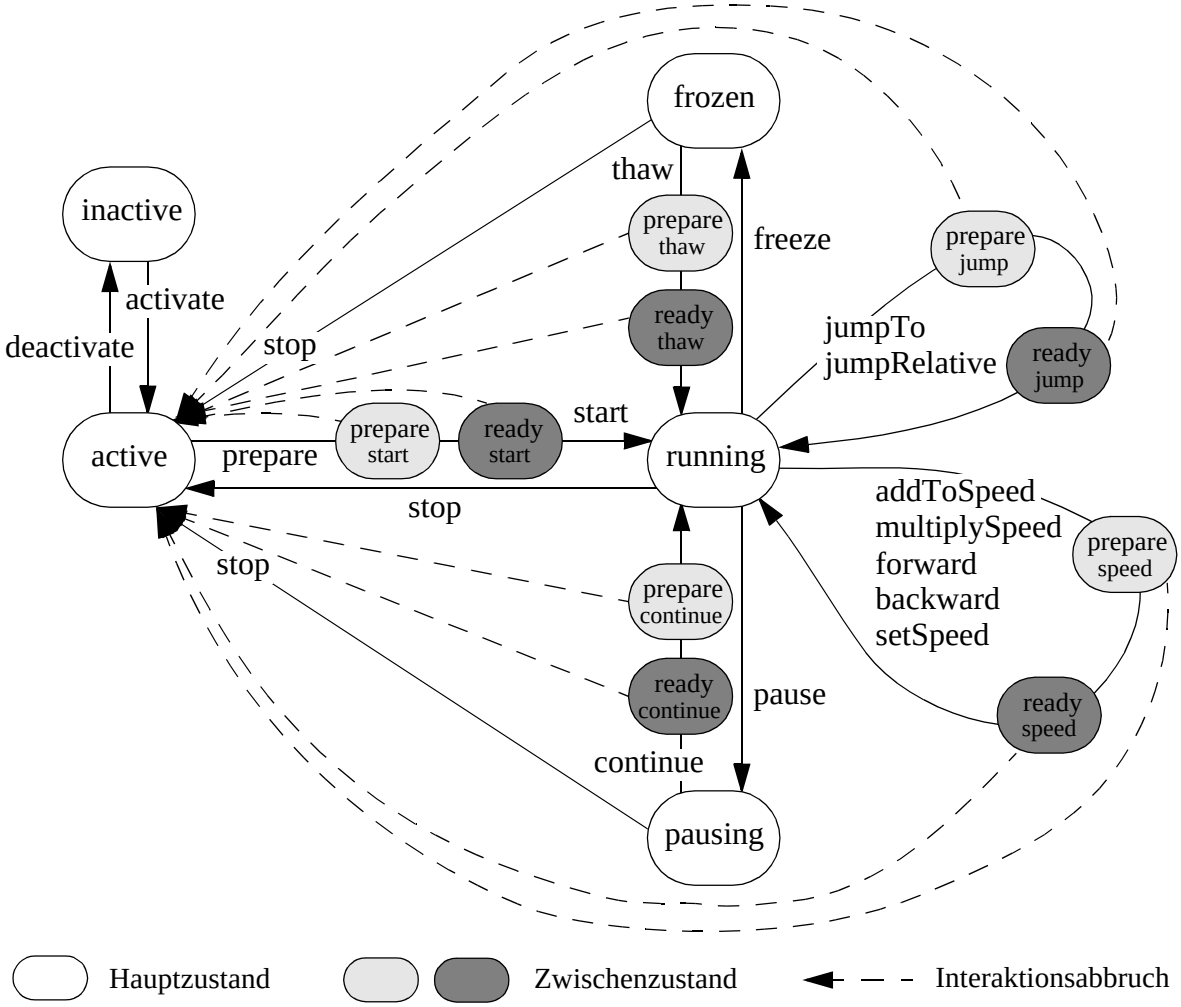


Abb. 8-11 : Projektor-Zustandsgraph

tion durch den Projektor entsprechend dem letzten Hauptzustand fortgeführt. Dabei kann es passieren, daß der Projektor laut Ablaufplan gestoppt werden muß, ehe der nächste Hauptzustand angenommen werden kann. In diesem Fall wird die Interaktion abgebrochen und der Projektor geht in den Zustand *active* über. Immer wenn ein Projektor in den Zustand *active* übergeht, gibt er den Speicher für seine Mediendaten frei.

8.6.3 Synchronisation kontinuierlicher Medien

Bei der Präsentation kontinuierlicher Medien wird eine einfache Intrastromsynchronisation verwendet. In Abhängigkeit des Startzeitpunkts s_K^o des Projektors *o*, der effektiven Präsentationsgeschwindigkeit v^o, der natürlichen Präsentationsrate r_0^o, der effektiven Präsentationsrate r^o

und des effektiven Ratenfaktors Δu^o wird der nominale Ausspielzeitpunkt t_u^o einer zu präsentierenden Dateneinheit mit der Sequenznummer u berechnet als:

$$t_u^o = s_K^o + \frac{1}{r^o}\left\lfloor \frac{u}{\Delta u^o} \right\rfloor = s_K^o + \frac{1}{r^o}\left\lfloor \frac{r^o u}{v^o r_0^o} \right\rfloor \qquad (8\text{-}9)$$

Mit Hilfe der nominalen Ausspielzeit wird ein Datenstrom entsprechend folgendem Algorithmus verarbeitet:

- Eine Dateneinheit u wird, sobald sie zum Ausspielen vorbereitet ist, zum frühesten Zeitpunkt $t \geq t_u^o$ ausgespielt.

- Die verfügbare Dateneinheit u', für welche die Differenz $t_{u'}^o - t$ den kleinsten positiven Wert annimmt, wird zum Zeitpunkt $p_{u'} = \max\{t_{u'}^o - 1/r^o, t\}$ vorbereitet. Ist keine Dateneinheit verfügbar, die diese Bedingung erfüllt, wird unter den noch nicht ausgespielten verfügbaren Dateneinheit die Dateneinheit u', für welche die Differenz $t_{u'}^o - t$ den kleinsten negativen Wert annimmt, unmittelbar vorbereitet. Es wird $u = u'$ gesetzt.

Dieser Algorithmus sorgt dafür, daß Dateneinheiten nicht früher als zu ihrem nominellen Ausspielzeitpunkt präsentiert werden und daß als nächstes immer die Dateneinheit ausgespielt werden kann, welche laut nominalen Ausspielzeiten an der Reihe ist und falls diese Dateneinheit nicht vorliegt, die letzte vorliegende Dateneinheit ausgespielt wird. Dementsprechend erfolgt beim Ausspielen eines kontinuierlichen Datenstroms eine Anpassung an Kommunikations-, Dekodier- oder Ausspielverzögerungen. Eine Interstromsynchronisation zwischen kontinuierlichen Medien wird nicht vorgenommen.

8.6.4 Steuerung des Medientransfers

Zum Transfer von Tiempo-Dokumenten und der darin enthaltenen Medien wird ein Kommunikationssystem benötigt, das es ermöglicht, diskrete Medien und kontinuierliche Medien vom Server zum Präsentationsterminal zu transferieren. Da kontinuierlichen Medien während des Ausspielens als isochrone Datenströme transferiert werden sollen, sind entsprechende Mechanismen zur Stromsynchronisation erforderlich.

In den letzten Jahren wurden zahlreiche Multimedia-Middleware Systeme entwickelt, die es erlauben Datenströme zu transferieren, zu synchronisieren und zu verarbeiten. Ein Beispiel für ein solches System ist CINEMA [BDH+93, Bart98]. Der Vorteil dieser Systeme ist, daß die Konfiguration [Bart96] und Synchronisation [RoHe95, Helb96] von Medienströmen relativ einfach zu bewerkstelligen ist. Bisher hat jedoch kein System eine breite Verfügbarkeit erreicht, insbesondere nicht im Umfeld des Internets. Des weiteren unterstützen diese Systeme meistens nicht die Kommunikation diskreter Medien, wodurch im Kontext von Multimedia-Dokumenten ein zweiter Kommunikationsmechanismus erforderlich ist. Für diese Arbeit wurde deshalb keines dieser Systeme verwendet, sondern auf standardisierten Protokollen für das Internet bzw. WWW aufgesetzt.

Im Tiempo-System wird für den Transfer von Dokumenten und diskreten Medien HTTP [FGM+97], zum Transfer von kontinuierlichen Medien RTP [SCF+96, Schu95] in Kombination mit RTSP [SRL97] eingesetzt. In Abbildung 8-12 ist die für diese Arbeit relevante Internet-Protokollhierarchie dargestellt.

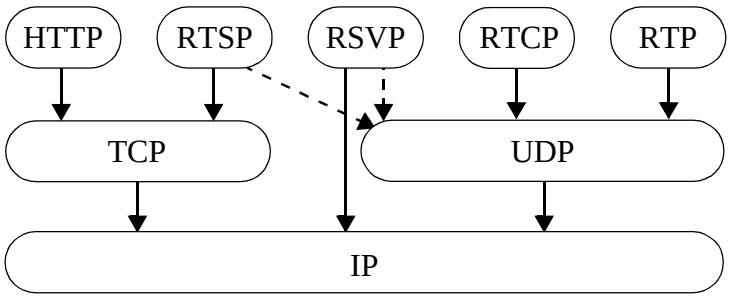


Abb. 8-12 : Internet-Protokollhierarchie

Für den Transfer von diskreten Medien empfielt sich aufgrund seiner Verbreitung und Eigenschaften die Verwendung von HTTP. HTTP setzt auf dem verbindungsorientierten TCP [Tane88] auf, und ermöglicht somit einen relativ sicheren Transfer im Hinblick auf Datenverluste. Die *prepare*-Aktionen im Ablaufplan werden auf HTTP GET-Nachrichten abgebildet.

Zur Kommunikation kontinuierlicher Medien mit einem garantierten QoS könnte RSVP [ZDE+93], SRP [AHS90] oder ST-II [Topo90] verwendet werden. Da wir in dieser Arbeit die

Kommunikation von Medienströmen ohne garantierten QoS im Zusammenhang mit adaptiven Dokumenten untersuchen, betrachten wir diese Protokolle jedoch nicht näher.

Zum Stromtransfer kontinuierlicher Medien wird RTP eingesetzt, das im allgemeinen in Verbindung mit UDP [Tane88] verwendet wird, jedoch sowohl auf verbindungslosen wie verbindungsorientierten Transportprotokollen aufsetzen kann. RTP ist ein Ende-zu-Ende Protokoll, das typischerweise als Teil einer Anwendung und nicht als Teil des Betriebssystemkerns implementiert wird. RTP ist ein Echtzeitprotokoll, das keine Reservierungen vornimmt und somit keine QoS-Garantien liefert.

Um ein Paket Mediendaten zu senden, formatiert ein Server die Daten für die Paketierung, fügt einen medienspezifischen Paketkopf hinzu und schickt das Paket über UDP ins Netzwerk, entweder zu einer Multicast-Gruppe oder zu einem einzelnen Empfänger. Ein RTP-Paketkopf enthält u.a. einen payload-Typ, der die im Paket verwendete Mediakodierung identifiziert, eine Sequenznummer, die sich von einem Paket zum nächsten erhöht und zur Erkennung von Paketverlusten und zur Wiederherstellung der Paketreihenfolge verwendet wird, sowie eine Zeitmarke, die mit der Medienrate hochgezählt wird und zeigt wann eine Dateneinheit generiert wurde. Als RTP payload-Formate wurden zahlreiche Video- und Audio-Formate definiert, wie beispielsweise H.263 [Zhu97], JPEG [BFF+96], MPEG [HFG+98].

RTCP ist das Kontrollprotokoll für RTP. Mediensender und -empfänger senden periodisch RTCP-Pakete. Jedes RTCP-Paket enthält einen Senderbericht oder Empfängerbericht. Senderberichte werden von Nutzern verschickt, die Mediendaten senden. Sie beschreiben die Datenmenge, die gesendet wurde, und enthalten eine Korrelation zwischen RTP-Zeitmarken und der Echtzeit, um beim Empfänger die Synchronisation zwischen den Medien zu ermöglichen. Anhand der Empfängerberichte von Nutzern, die Mediendaten empfangen, kann ein Sender den Stromtransfer so steuern, daß der gewünschte periodische Transfer stattfindet.

RTP ist ein Multicast-fähiges Protokoll, d.h. ein Sender kann einen Datenstrom gleichzeitig an mehrere Empfänger schicken. Deshalb werden QoS-Beobachtungen und -Steuerungen von den Sendern vorgenommen. Das in dieser Arbeit betrachtete Verteilungsszenario ist genau umgekehrt. Wir haben ein Präsentationsterminal zu dem von vielen Servern Daten transferiert werden und der Empfänger führt QoS-Beobachtungen und -Steuerungen durch. Es ist daher ein

weiteres Protokoll zur Steuerung der Server durch das Präsentationsterminal nötig. Im Tiempo-System wird hierzu das standardisierte Protokoll RTSP eingesetzt, das die Steuerung des Transfers von gespeicherten oder live erzeugten Mediendaten zu Multicast-Gruppen oder einzelnen Empfängern erlaubt. RTSP ist wie RTP auf der Applikationsebene angesiedelt und kann auf TCP oder UDP aufsetzen.

In Abbildung 8-13 ist das Schema zu sehen, wie eine Dokumentpräsentation mit RTSP gesteuert werden kann. Das Dokument wird beispielsweise mit HTTP von einem Web-Server zum Präsentationsterminal (Client) transferiert. Dann wird der Ablaufplan erstellt und der Transfer der enthaltenen kontinuierlichen Medien über RTSP gesteuert. Durch eine *setup*-Nachricht wird der Stromtransfer eines Mediums initialisiert, hierbei wird eine Sitzung aufgebaut. Eine *play*-Nachricht startet den Stromtransfer einer Sequenz von Dateneinheiten zu einem definierten Zeitpunkt über RTP. Während des Transfers können Steuernachrichten zur Veränderung der Transferrate oder Geschwindigkeit abgesetzt werden, sowie der Transfer durch eine *pause*-Nachricht unterbrochen werden. Die Sitzung wird durch eine *teardown*-Nachricht beendet.

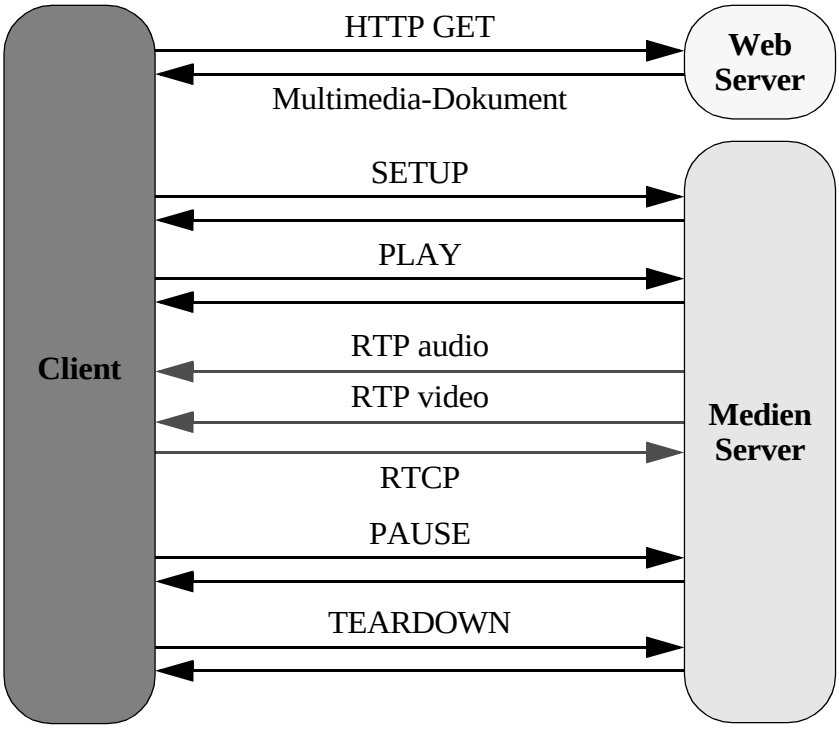


Abb. 8-13 : Steuerung einer Dokument-Präsentation mit RTSP

Auf solche RTSP-Nachrichten können sowohl die Interaktionen des Tiempo-Modells als auch die durch Adaptionen erforderlichen Anpassungen des Stromtransfers unmittelbar abgebildet werden. Die *prepare_stream*-Aktionen im Ablaufplan werden beispielsweise durch *setup*- und *play*-Nachrichten realisiert.

8.7 Leistungsmessungen

Zur Bewertung des Ansatzes wurden Messungen durchgeführt. Tabelle 8-2 zeigt Ablaufplanungsdauern ohne daß Prioritäten aus Dienstgütebereichen berücksichtigt wurden. Hierbei ist horizontal die Anzahl der Projektoren im Dokument aufgeführt und vertikal die Anzahl der betrachteten eindeutigen Ereignisgraphen. Wie zu sehen ist, ergibt sich ein nahezu linearer Anstieg bei den Ablaufplanungsdauern.

		Anzahl Projektoren					
		1	2	5	10	20	40
Anzahl Ereignisgraphen	1	20 ms	30 ms	52 ms	91 ms	118 ms	180 ms
	2	40 ms	63 ms	110 ms	205 ms	250 ms	334 ms
	5	68 ms	110 ms	220 ms	324 ms	410 ms	480 ms
	10	101 ms	170 ms	380 ms	582 ms	693 ms	820 ms

Tabelle 8-2: Ablaufplanungsdauern ohne Berücksichtigung von Prioritäten

Tabelle 8-3 zeigt die Ablaufplanungsdauern wenn das polynomiale Optimierungsverfahren angewendet wird und die Spezifikation viele Dienstgütebereiche mit Prioritäten enthält. Hier ergibt sich nun im Bezug auf die Anzahl der Projektoren ein eher quadratischer Anstieg. Die Ablaufplanungsdauern sind jedoch auch bei größeren Präsentation akzeptabel.

Tabelle 8-4 zeigt die Ablaufplanungsdauern wenn das NP-vollständige Optimierungsverfahren angewendet wird und die Spezifikation viele Dienstgütebereiche mit Prioritäten enthält. Bei diesem Optimierungsverfahren ergibt sich im Bezug auf die Anzahl der Projektoren ein drastischer Anstieg der Ablaufplanungsdauern. Sie überschreiten schon bei fünf Projektoren und fünf Graphen die Sekundengrenze, bei mehr als 40 Projektoren benötigt man schon bei nur einem

		Anzahl Projektoren					
		1	2	5	10	20	40
Anzahl Ereignisgraphen	1	25 ms	28 ms	61 ms	110 ms	201 ms	480 ms
	2	44 ms	49 ms	123 ms	250 ms	380 ms	934 ms
	5	72 ms	101 ms	320 ms	634 ms	1050 ms	2470 ms
	10	115 ms	240 ms	645 ms	903 ms	2010 ms	4960 ms

Tabelle 8-3: Ablaufplanungsdauern mit Berücksichtigung von Prioritäten entsprechend dem polynomialen Ansatz

Ereignisgraph zirka zehn Sekunden. Dementsprechend ist dieses Optimierungsverfahren für größere Präsentation nicht geeignet.

		Anzahl Projektoren					
		1	2	5	10	20	40
Anzahl Ereignisgraphen	1	30 ms	38 ms	201 ms	601 ms	2.1 s	8.5 s
	2	47 ms	75 ms	580 ms	1.5 s	5.7 s	22 s
	5	80 ms	163 ms	1.1 s	4.0 s	12 s	40 s
	10	150 ms	500 ms	2.3 s	8.3 s	32 s	92 s

Tabelle 8-4: Ablaufplanungsdauern mit Berücksichtigung von Prioritäten entsprechend dem NP-vollständigen Ansatz

8.8 Diskussion und Bewertung

Das vorgestellte Tiempo-Ablaufplanungsverfahren integriert einen effizienten Algorithmus zur Bestimmung eines ressourcenoptimalen Ablaufplans. Von den beiden vorgestellten Optimierungsverfahren, die auf diesem Algorithmus aufsetzen, ist das, welches die Summe der Prioritäten optimiert, nur für kleine Präsentation anwendbar, da für große Präsentationen die Ablaufplanungsdauer zu groß wird. Demgegenüber zeigt das Optimierungsverfahren, das anhand von Prioritäten entscheidet in welcher Reihenfolge flexiblen Attributen Werte zugewiesen werden,

einen moderaten Anstieg der Ablaufplanungsdauer und ist somit auch für große Präsentationen geeignet.

Wird die Ablaufplanungsdauer bei großen Präsentation zu lang, kann nur für einen eingeschränkten Zeitraum eine Ressourceneinplanung vorgenommen werden oder die Optimierung nach einer vorgegebenen Ablaufplanungsdauer einfach ausgeschaltet werden. Dies ist bei beiden Optimierungsverfahren ohne Probleme möglich, da eine ressourcenoptimale Variablenbelegung einen validen Ablaufplan ergibt und bei jeder Iteration der Optimierungsverfahren eine solche Belegung entsteht.

Das Tiempo-Ablaufplanungsverfahren ist für Präsentationsumgebungen geeignet, die eine Ressourcenreservierung ermöglichen, und für Umgebungen, die eine Ressourcenzuteilung nach dem Best-Effort Prinzip durchführen. Der Unterschied besteht lediglich darin, wie die Aktionen im Ablaufplan abgebildet werden. Bei einer Präsentationsumgebung mit Ressourcenreservierung werden jedoch weniger Adaptionen erforderlich sein. Wobei – wie schon erwähnt – auch hier Ressourcenengpässe durch Interaktionen auftreten können.

Beim Vergleich mit dem Ablaufplanungsverfahren von CHIMP [CPS96] muß berücksichtigt werden, daß dieses keine Optimierung auf Attributebene durchführt, d.h. es verhält sich so, wie der Algorithmus zur Bestimmung des ressourcenoptimalen Ablaufplans. Vergleicht man diese beiden Konzepte miteinander, stellt man fest, daß in CHIMP aufgrund des verwendeten Kürzeste-Pfad Algorithmuses zur Bestimmung einer Variablenbelegung nicht immer eine Belegung entsteht, die einen minimalen Ressourcenverbrauch impliziert. Dadurch muß bei CHIMP zur Einplanung von positionsunabhängigen Ressourcenanforderungen der Kürzeste-Pfad Algorithmus wiederholt angewendet werden, was zu längeren Planungsdauern führt. Beim Tiempo-Ablaufplanungsverfahren hingegen kann die komplette Einplanung positionsunabhängiger Ressourcen in einem Durchlauf bestimmt werden.

Da Firefly [BuZe92, BuZe93] und Mbuild [HSR92, HSR94] keine Ressourcen bei der Ablaufplanung betrachten, ist kein Vergleich möglich. Ebenso ist kein Vergleich mit der Ablaufplanung von MODE [BHL91] möglich, da Details nicht veröffentlicht wurden.

9 Architektur erweiterbarer Dokumentensysteme

In diesem Abschnitt wird ein Architekturkonzept für erweiterbare Dokumentensysteme vorgestellt. Ziel des Architekturkonzepts war es, eine einfache Anpassung der Funktionen eines Dokumentensystems bei Änderungen der Dokumentensprache zu ermöglichen. Basierend auf diesem Architekturkonzept wurde das Tiempo-Dokumentensystem prototypisch implementiert [Maye95, Fran95, Fisc95, Wira99a], um die in dieser Arbeit vorgestellten Konzepte zu evaluieren und verifizieren.

Im Abschnitt 9.1 werden Erweiterbarkeitsaspekte von Multimedia-Dokumentensystemen erörtert. In Abschnitt 9.2 wird die funktionale Architektur des Tiempo-Dokumentensystems vorgestellt. Anschließend werden das objektorientierte Modell, welches eine einfache Erweiterung ermöglicht, sowie der Erweiterungsprozeß erörtert.

9.1 Erweiterbarkeitsaspekte

Multimedia-Dokumentensprachen sind einer ständigen Weiterentwicklung unterworfen, da sich einerseits durch die stetig wachsende Leistungsfähigkeit von Computern und Netzwerken neue Möglichkeiten bei der Gestaltung von Multimedia-Präsentationen eröffnen, andererseits aber auch Nutzer immer höhere Anforderungen an das Design und die Gestaltung von Multimedia-Präsentationen stellen. Im Bezug auf Multimedia-Dokumentensysteme ergibt sich damit die Anforderung, daß sie einfach entsprechend den Erweiterungen von Dokumentensprachen angepaßt werden können müssen. Einfach bedeutet hierbei, daß neue Funktionalitäten mit geringem Aufwand hinzugefügt werden können.

Prinzipiell kann zwischen zwei Arten von Erweiterbarkeit unterschieden werden:

- Unter **Medienerweiterbarkeit** versteht man die Möglichkeit zur Integration von Sprachelementen für neue Medientypen.
- Unter **Konzepterweiterbarkeit** versteht man die Möglichkeit zur Integration von Spracherweiterungen für die Spezifikation von räumlichen, zeitlichen, interaktiven oder adaptiven Dokumentaspekten.

Vergleicht man diese beiden Arten der Erweiterbarkeit, stellt man fest, daß die Konzepterweiterbarkeit wesentlich schwieriger zu realisieren ist als die Medienerweiterbarkeit. Dies liegt daran, daß sich in der Regel die Semantik jedes Medienobjekts durch einen abstrakten Datentyp mit einer einheitlichen Schnittstelle kapseln läßt und die Integration eines neuen Medientyps somit meist keine Änderungen an funktionalen respektive algorithmischen Komponenten eines Dokumentensystems erfordern.

Mit Konzepten ist dies in der Regel nicht so einfach. Möchte man beispielsweise im Tiempo-Modell die Projektion von Datenräumen so erweitern, daß man mittels eines Attributs beschreiben kann, daß der Datenraum mehrmals hintereinander projiziert werden soll, muß man die Konsistenzprüfungs- und Ablaufplanungsverfahren erweitern. Algorithmen so zu entwerfen und zu implementieren, daß sie einfach ausgetauscht bzw. beliebig konfiguriert werden können ist jedoch komplex. Lösungsansätze für dieses Problem werden im MAVA-Projekt [HaRo00] untersucht. In diesem Projekt wird ein Metadokumentenmodell entwickelt, mit dem man beliebige Dokumentenmodelle beschreiben kann, die jeweils aus Medienobjekten und Konzepten zu deren Verknüpfung bestehen. Da in diesem Metadokumentenmodell Konzepte explizit durch sogenannte Manager modelliert werden, die einheitliche Schnittstellen besitzen, wird es u.a. möglich, Konzepte in einem Dokumentensystem auszutauschen. In dieser Arbeit wird nur die Medienerweiterbarkeit betrachtet.

9.2 Funktionale Architektur

Auf der Grundlage der entwickelten Konzepte wurde prototypisch das Tiempo-Dokumentensystem realisiert [Wira99a], welches die Erstellung und Präsentation verteilter Tiempo-Dokumente ermöglicht. Das Tiempo-Dokumentensystem integriert sowohl die Funktionalität zur Spezifikation als auch zur Präsentation von Tiempo-Dokumenten. Auf diese Weise können Dokumente oder Dokumentteile während des Spezifikationsprozesses unmittelbar getestet werden. Abbildung 9-1 zeigt die Architektur des Tiempo-Dokumentensystems, das aus folgenden funktionalen Komponenten besteht:

- In einem *Type Repository* werden Informationen über die Dokumentelementtypen des Tiempo-Dialekts verwaltet, aus denen sich ein Dokument zusammensetzen kann. Jeder Dokumentelementtyp wird durch ein Typobjekt repräsentiert, das beschreibt, welche

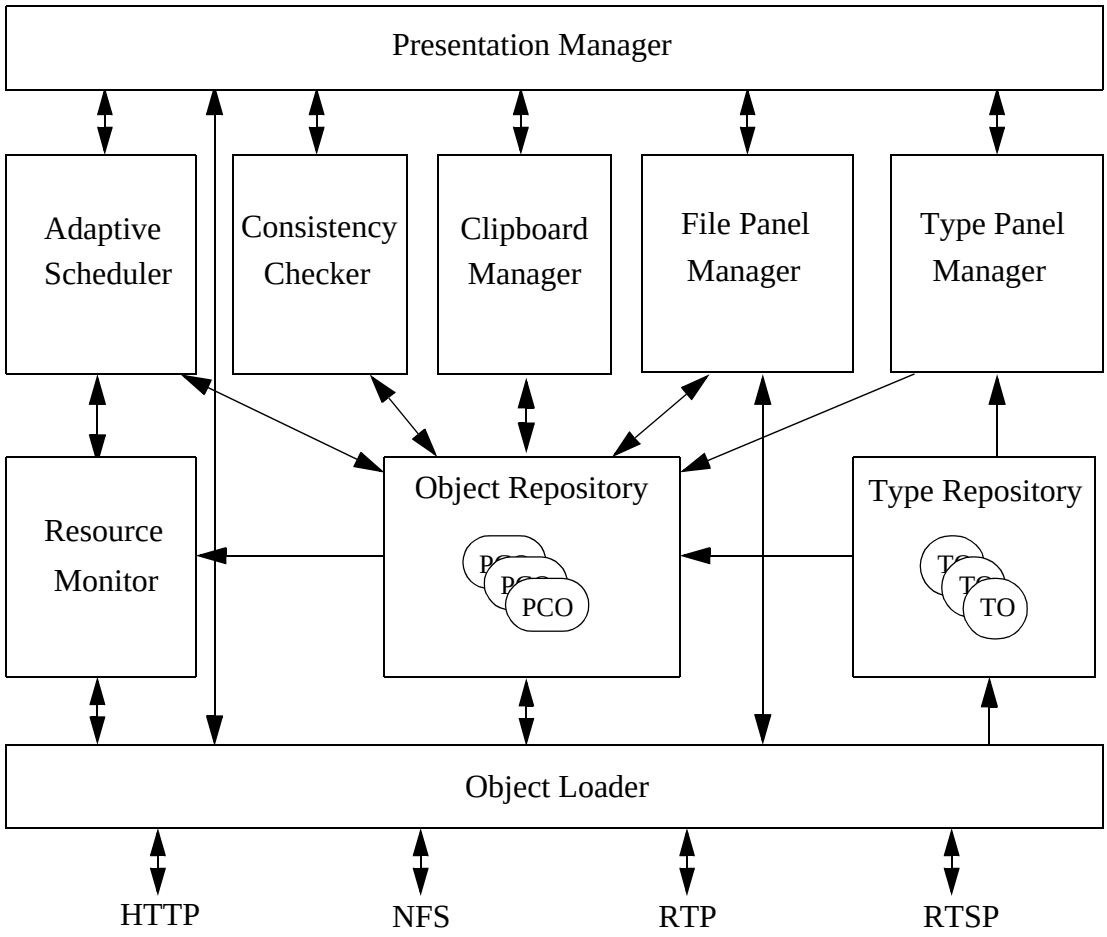


Abb. 9-1 : Tiempo Systemarchitektur

Dokumentelemente ein Element dieses Typs enthalten darf und welche Attribute es besitzen muß. Das Typobjekt enthält zudem Standardwerte für Attribute. Die Typobjekte repräsentieren somit die in einer Tiempo-DTD definierten XML-Elementtypen.

- Im *Object Repository* werden die Präsentationskontrollobjekte (PCO) von Dokumenten verwaltet. Für jeden Dokumentelementtyp der Tiempo-Sprache gibt es eine spezielle PCO-Klasse, welche die Funktionalität zur Interpretation von Dokumentelementen diesen Typs sowohl im Editor als auch zur Präsentation integriert. Ein PCO enthält auch die entsprechenden Informationen des Dokumentelements.

- Der *Object Loader* dient zum Transfer von Dokumenten und Mediendaten. Er bildet damit die Schnittstelle des Tiempo-Dokumentensystems zu Servern. Der Loader überprüft beim Laden eines Dokuments dessen Syntax. Des weiteren überprüft er anhand der Typobjekte

im *Type Repository*, ob ein geladenes Dokumentelement die erforderlichen Attribute besitzt. Für jedes im Dokument enthaltene Dokumentelement wird ein Präsentationskontrollobjekt im *Object Repository* erzeugt.

Im Tiempo-System wird auf verteilte Dokumente und statische Medien über NFS oder HTTP [FGM+97] zugegriffen. Der Stromtransfer kontinuierlicher Medien wird über das RTSP-Protokoll [SRL97] gesteuert. Der Stromtransfer selbst erfolgt mit RTP [SCF+96, Schu95].

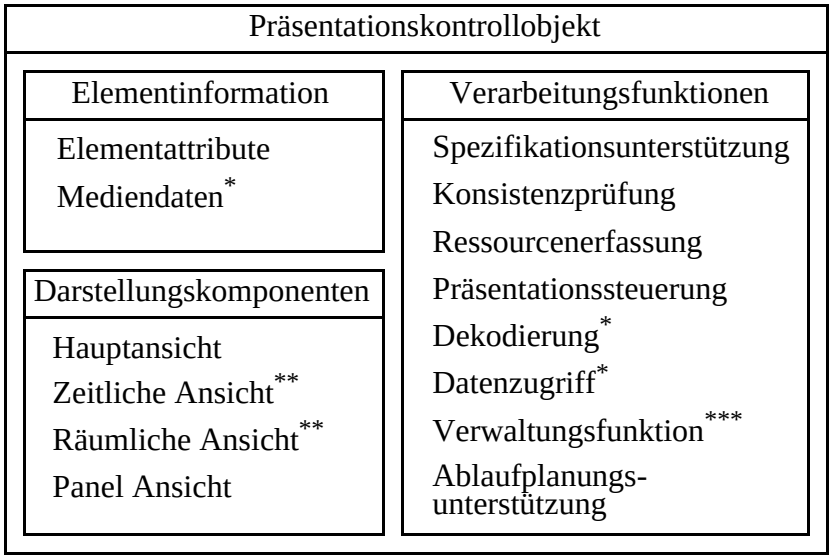
- Der *Type Panel Manager* dient zur Verwaltung der Type Selection Panels des Editors.
- Der *Clipboard Manager* verwaltet das Clipboard des Editors
- Der *File Panel Manager* dient zur Verwaltung der File Selection Panels des Editors.
- Der *Consistency Checker* organisiert die Konsistenzprüfung von Spezifikationen und Dokumenten, basierend auf dem in Abschnitt 5 beschriebenen Algorithmus.
- Der *Resource Monitor* sammelt Informationen über verfügbare und verwendete Ressourcen vom *Object Loader* und den PCOs und bereitet sie zur Verwendung durch den Ablaufplaner (Scheduler) auf.
- Der *Adaptive Scheduler* berechnet den Ablaufplan für Dokumente. Hierzu wird der in Abschnitt 8 erläuterte Ablaufplanungsalgorithmus eingesetzt. Die benötigten Ressourceninformationen werden vom *Resource Monitor* erfragt.
- Der *Presentation Manager* koordiniert die Aktivitäten der anderen Komponenten. Er initiiert das Laden von Dokumenten sowie die Berechnung des Ablaufplans. Der Präsentationsmanager arbeitet den vom Ablaufplaner berechneten Ablaufplan ab, indem er PCOs zu den jeweiligen Zeitpunkten entsprechende Nachrichten schickt.

Das Tiempo-Dokumentensystem wurde in C++ [Stro87] und X/Motif [GKK+92] unter dem Betriebssystem SUN Solaris und Linux implementiert.

9.3 Das Konzept der Präsentationskontrollobjekte

Ein Präsentationskontrollobjekt repräsentiert die gesamte Funktionalität, die zur Verarbeitung eines damit assoziierten Dokumentelements notwendig ist und zwar sowohl hinsichtlich der Dokumenterstellung wie auch der Dokumentpräsentation. Alle Präsentationskontrollobjektklassen gehen durch Vererbung auseinander hervor. Da es für jeden Dokumentelementtyp eine separate Präsentationskontrollobjektklasse gibt, hat der Klassenbaum der Präsentationskontrollobjekte die gleiche Struktur wie der Vererbungsbaum der Dokumentelementtypen der Tiempo-Sprache bzw. eines Tiempo-Dialekts. Durch diesen objektorientierten Ansatz aus Kapselung von Funktionalität in Kombination mit Vererbung wird eine Implementierungsstruktur ermöglicht, die eine einfach Medienerweiterbarkeit unterstützt.

Präsentationskontrollobjekte haben die in Abbildung 9-2 dargestellte Model-View-Controller-Struktur [Fowl97]:



* nur bei Medien-PCO ** nur bei Projektor-PCO *** nur bei Multimedia-Projektor-PCO

Abb. 9-2 : Struktur der Präsentationskontrollobjekte

- Die **Elementinformation** (Model) ist äquivalent zur Information, welche in einem Dokument über das durch das PCO repräsentierte Dokumentelement enthalten ist. Sie besteht

aus Elementattributen und – falls es sich um ein Medienelement handelt – eventuell aus im Dokument enthaltenen Mediendaten.

- Die **Darstellungskomponenten** (View) ermöglichen die Darstellung des Dokumentelements im Dokumenteditor und die Präsentation der Elementdaten während einer Dokumentpräsentation. Darstellungskomponenten zur Präsentation von Mediendaten haben nur PCOs die Projektoren repräsentieren.

- Die **Verarbeitungsfunktionen** (Control) ermöglichen die Verarbeitung des durch das PCO repräsentierten Dokumentelements im Rahmen des Spezifikationsprozesses, der Konsistenzprüfung, der adaptiven Ablaufplanung und Präsentation. Jede PCO-Klasse enthält dabei genau die Funktionalität, welche für die Verarbeitung des zugehörigen Dokumentelementtyps erforderlich ist:

  - Die Spezifikationsunterstützung ermöglicht die Verarbeitung bei der interaktiven graphischen Spezifikation im Editor.

  - Eine Verwaltungsfunktion sorgt bei PCOs, die Dokumentelementtypen repräsentieren, welche andere Dokumentelemente enthalten können (z.B. Multimedia-Projektoren), für die Verwaltung der PCOs enthaltener Elemente.

  - PCOs von Medienelementtypen enthalten – falls die Daten nicht direkt im Dokument enthalten sind – eine Datenzugriffsfunktionalität, die es ermöglicht, mittels eines entsprechenden Kommunikationsprotokolls auf die extern gespeicherten Daten zuzugreifen.

  - PCOs von Medienelementtypen enthalten eine Dekodierungsfunktionalität, welche es ermöglicht, Dateneinheiten in eine präsentierbare Form zu konvertieren.

  - Die Konsistenzprüfung realisiert den in Abschnitt 5 vorgestellten Konsistenzprüfungsalgorithmus für den repräsentierten Dokumentelementtyp.

  - Die Ablaufplanungsunterstützung liefert dem Ablaufplaner die zur Ablaufplanung benötigten Informationen über das Dokumentelement.

- Zur Bereitstellung der Ressourcenanforderungsinformationen dient die Ressourcenerfassungsfunktion.

- Die Präsentationssteuerung dient sowohl beim Spezifikationsprozeß als auch bei der Präsentation zur Umsetzung der Nachrichten vom Präsentationsmanager.

In Bezug auf die in Abbildung 9-1 dargestellten funktionalen Komponenten implementiert ein PCO die Funktionalität, welche zur Verarbeitung des repräsentierten Dokumentelements erforderlich ist. Die funktionalen Komponenten übernehmen die Ausführungskontrolle für die jeweilige Funktion.

9.4 Das Objektmodell des Tiempo-Dokumentensystems

Abbildung 9-3 zeigt als UML-Klassendiagramm [GRJ99] wie das PCO-Konzept in das Tiempo-Dokumentensystem integriert wurde. Es gibt im wesentlichen drei Klassenhierarchien:

- Präsentationskontrollobjektklassen (siehe oben),
- Darstellungsklassen (siehe links unten)
- Kommunikationsklassen (siehe rechts unten).

Durch die Aufteilung der benötigten Funktionalität auf mehrere Klassenhierarchien können objektorientierte Konzepte wie Vererbung und Polymorphismus dediziert eingesetzt werden, was den Implementierungsaufwand für Erweiterungen stark reduziert.

PCO-Klassen, die von der Klasse *PCOMedia* abgeleitet werden, repräsentieren Medien. Um auf externe Daten eines Mediums zuzugreifen, kann ein solches PCO die Instanz einer Kommunikationsklasse integrieren, die von der Klasse *TAMChannel* abgeleitet ist. Im Tiempo-Dokumentensystem sind alle verwendeten Kommunikationsprotokolle (HTTP, RTP, RTSP, NFS) als eine eigene Kommunikationsklasse realisiert. Die Instanz einer solchen Klasse repräsentiert die Schnittstelle zu einem Kommunikationskanal, mit dessen Hilfe man über das jeweilige Protokoll mit einem Server kommunizieren kann. Alle Instanzen von Kommunikationsklassen werden durch den *Object Loader* koordiniert. Somit ist das Management einer Stromverbindung aus Sicht eines PCOs sehr einfach.

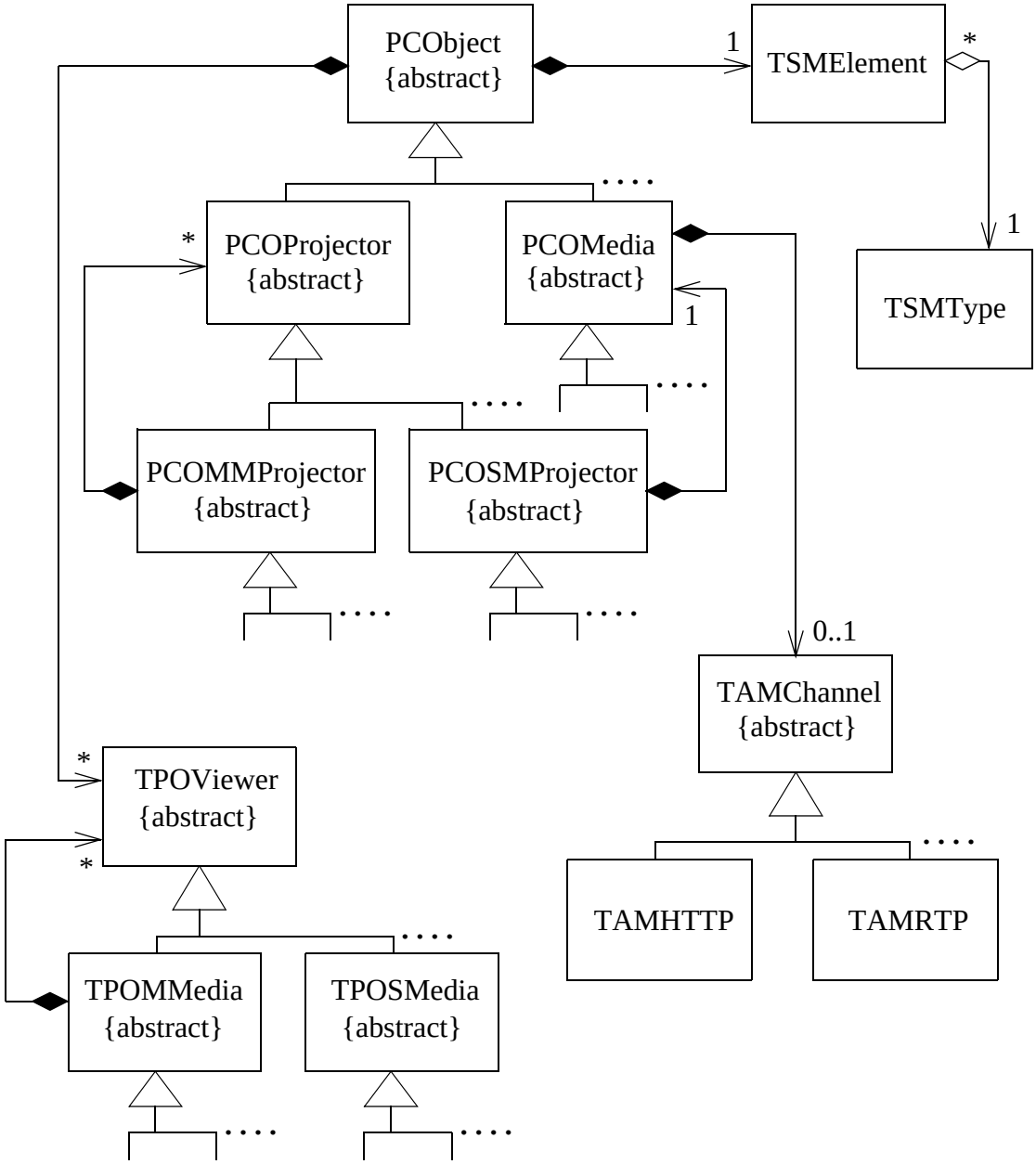


Abb. 9-3 : Struktur des Tiempo-Objektmodells

Um bei der Entwicklung von Darstellungselementen das Vererbungskonzept gewinnbringend einsetzen zu können, wie beispielsweise bei der Berechnung von Ausgabemasken, werden alle Darstellungselemente in einer eigenen Klassenhierarchie *TPOViewer* implementiert. Jede Instanz einer PCO-Klasse kann mehrere Instanzen von Darstellungsklassen enthalten.

Jede Instanz einer PCO-Klasse enthält genau eine Instanz der Klasse *TSMElement*, welche die Information über das mit dem PCO assoziierte Dokumentelement beinhaltet. Jeder Dokument-

elementtyp der Tiempo-Sprache bzw. eines Tiempo-Dialekts wird durch genau eine Instanz der Klasse *TSMType* beschrieben. Um Standardwerte von Dokumentelementattributen bereitstellen zu können, ist jede Instanz der Klasse *TSMElement*, in Abhängigkeit des Dokumentelementtyps den sie repräsentiert, mit der entsprechenden Instanz der Klasse *TSMType* verbunden.

Alle Instanzen von PCO-Klassen werden entsprechend dem Object-Factory Konzept [Fowl97] in der Implementierung des Tiempo-Dokumentensystem an einer einzigen Stelle generiert und zwar im *Type Repository*.

9.5 Erweiterungsprozeß für Medienobjekte

In diesem Abschnitt wird erläutert, wie ein Dokumentensystem, das auf dem PCO-Konzept basiert, um neue Medienobjekte erweitert werden kann. Beispiele für derartige Erweiterungen sind die Integration eines neuen Videoformats oder eines neuen Interaktionselements.

Um einen Tiempo-Dialekt und das Tiempo-Dokumentensystem um eine neue Medienart zu erweitern, sind entsprechende Dokumentelementtypen zu definieren, es müssen geeignete Präsentationskontrollobjekte implementiert werden und die Konfiguration des Editors ist anzupassen.

9.5.1 Erweiterung eines Tiempo-Dialekts (Tiempo-DTD)

Es muß ein neuer Medienelementtyp in den Vererbungsbaum des Tiempo-Dialekts eingefügt werden, der die neue Medienart beschreibt. Falls kein geeigneter Projektor zur Präsentation der neuen Medienart existiert, muß ebenfalls ein neuer Projektorelementtyp eingefügt werden. Handelt es sich bei der neuen Medienart um ein Interaktionsmedium, muß – falls nicht schon vorhanden – ein geeignetes Interaktionsmanagerelement in den Vererbungsbaum eingefügt werden.

Alle Erweiterungen eines Tiempo-Dialekts können durch den Tiempo-Editor unmittelbar verarbeitet werden, solange die neuen Dokumentelementtypen von Dokumentelementtypen im Vererbungsbaum abgeleitet werden.

9.5.2 Implementierung geeigneter Präsentationskontrollobjekte

Für jeden neuen Dokumentelementtyp muß ein entsprechendes Präsentationskontrollobjekt implementiert und für das Dokumentensystem bereitgestellt werden. Durch die Vererbung müssen hierbei lediglich Funktionen zur Darstellung und Dekodierung der neuen Medienart sowie zur Erfassung des Ressourcenbedarfs implementiert werden. Die anderen Funktionen sind bereits in Basisklassen implementiert. Hierbei muß, abhängig von der Art der PCO-Klasse, eine entsprechende Schnittstelle in Form von Methoden implementiert werden.

Für eine PCO-Klasse, die ein Medienelement repräsentiert, müssen folgende Methoden implementiert werden:

- *init*: Zur Generierung der Darstellungsinstanzen und Kommunikationsinstanzen
- *prepare*: Initialisierung der durch das PCO repräsentierten Daten. Bei einem kontinuierlichen Medium wird beispielsweise der Stromtransfer initialisiert.
- *destroy*: Beendigung des Zugriffs auf die durch das PCO repräsentierten Daten. Bei einem kontinuierlichen Medium wird beispielsweise der Stromtransfer terminiert.
- *deliverDataUnit*: Liefert eine spezifische Dateneinheit (z.B. Video-Frame, Audio-Sample) dekodiert zurück.
- *deliverAttributes*: Liefert die Attribute des durch das PCO repräsentierten Medienelements zurück. Darin sind auch die von Ablaufplaner benötigten plattformunabhängigen Ressourceinformationen (z.B. die durchschnittliche Größe einer Dateneinheit) enthalten.
- *deliverResourceNeeds:* Wird vom Ablaufplaner aufgerufen, um die dynamischen Ressourcenanforderungen des Medienelements abzufragen (z.B. die benötigten CPU-Zyklen zur Dekodierung einer Dateneinheit).

Für eine PCO-Klasse, die ein Projektorelement repräsentiert, müssen die folgenden Methoden implementiert werden:

- *init*: Zur Generierung der enthaltenen Darstellungskomponenten

- *deliverResourceNeeds:* Wird vom Ablaufplaner aufgerufen, um die Ressourcenanforderungen des Projektorelements zu erfragen, wie beispielsweise die benötigten CPU-Zyklen zum Ausspielen einer Dateneinheit.

Existiert keine geeignete Darstellungskomponente für die räumliche Darstellung respektive das Ausspielen von Dateneinheiten, ist eine neue Darstellungsklasse in den Vererbungsbaum einzufügen, die folgende Methoden implementiert:

- *init*: Zur Initialisierung der Darstellungskomponente.

- *presentDataUnit*: Wird zur Präsentation einer Dateneinheit aufgerufen.

Für eine PCO-Klasse, die einen Interaktionsmanager repräsentiert, muß lediglich eine Methode implementiert werden:

- *deliverState:* Wird vom Ablaufplaner aufgerufen, um den Status eines Interaktionsmanagers abzufragen.

Wie bereits erwähnt, wurde das Tiempo-Dokumentensystem prototypisch in C++ implementiert. Da für C++ keine plattformunabhängige direkt ausführbare Repräsentation existiert, wie beispielsweise der Byte-Code der Programmiersprache Java [Flan99], müssen neue PCO- und Darstellungsklassen unter dem jeweiligen Betriebssystem kompiliert und gelinkt werden. Da die meisten modernen Betriebssysteme die Möglichkeit bieten, dynamisch ladbare Objekte bzw. Bibliotheken zu erzeugen, ist dies aber nicht sehr aufwendig. Die neu implementierten Klassen sind so zu kompilieren und linken, daß eine dynamisch ladbare Bibliothek entsteht. Des weiteren ist die – ebenfalls als dynamisch ladbare Bibliothek realisierte – PCO-Generierungsfunktion des *Type Repositories* um die neuen PCO-Klassen zu erweitern, zu kompilieren und zu linken. Alle anderen Module des Dokumentensystems bleiben unverändert.

9.5.3 Konfiguration der Darstellung im Editor

Dem Editor muß in einer Konfigurationsdatei mitgeteilt werden, durch welches Icon ein neuer Dokumentelementtyp bzw. Instanzen des neuen Dokumentelementtyps im Editor dargestellt werden sollen.

9.6 Zusammenfassung

Die Architektur des Tiempo-Dokumentensystems integriert die Funktionalität zur Spezifikation von Dokumenten und zur Präsentation von Dokumenten. Dadurch können Dokumente oder Dokumentteile während des Spezifikationsprozesses getestet werden.

Das Konzept der Präsentationskontrollobjekte sorgt für eine Kapselung von Daten und Funktionen der Dokumentelementtypen der Tiempo-Sprache bzw. eines Tiempo-Dialekts. Durch die Vererbung von PCO-Klassen können einmal implementierte Funktionen in allen davon abgeleiteten PCO-Klassen verwendet werden. Diese beiden objektorientierten Konzepte schaffen die Grundlage für eine einfache Erweiterung von Tiempo-Dialekten und des Tiempo-Dokumentensystems um neue Medientypen.

10 Zusammenfassung und Ausblick

Bei der Präsentation verteilter Multimedia-Dokumente treten häufig unkontrollierte Qualitätsverluste durch fehlende Ressourcen auf. Zur Lösung dieses Problems wurden adaptive Multimedia-Dokumente vorgeschlagen, die Präsentationsalternativen enthalten, wodurch sie an unterschiedliche Ressourcensituationen angepaßt werden können.

Das in dieser Arbeit präsentierte Dokumentenmodell Tiempo integriert ausdrucksstarke Abstraktionen zur Spezifikation von Präsentationsalternativen. Durch Auswahlgruppen, welche die Spezifikation von Präsentationsalternativen auf Dokumentelementebene ermöglichen, können beliebig komplexe Dokumentteile als Alternativen deklariert werden. Durch den Einsatz von Dienstgütebereichen, die Präsentationsalternativen auf Attributebene beschreiben, können zeitlich flexible Dokumente spezifiziert werden. Durch die Möglichkeit dediziert Prioritäten und Auswahlpolitiken zuzuweisen, kann gesteuert werden, wie Präsentationsalternativen bei Ressourcenengpässen eingesetzt werden. Die flexiblen Projektionen des Tiempo-Modells erlauben es detailliert festzulegen, wie die Darstellung von Medien angepaßt werden soll. Indem Interaktionsmedien in Auswahlgruppen definiert werden oder ihre sensitive Phase durch Dienstgütebereiche flexibel spezifiziert wird, entstehen Multimedia-Dokumente, die adaptiv hinsichtlich der angebotenen Interaktionsmöglichkeiten sind. Durch den kombinierten Einsatz von Auswahlgruppen und Dienstgütebereichen können hochgradig adaptive Multimedia-Dokumente spezifiziert werden.

Adaptive Multimedia-Dokumente haben aufgrund der enthaltenen Präsentationsalternativen eine komplexere Struktur als herkömmliche Dokumente. Es werden daher geeignete Unterstützungstechniken für den Dokumenterstellungsprozeß und die Konsistenzsicherung benötigt.

Das in dieser Arbeit vorgestellte Visualisierungs- und Spezifikationskonzept für das Tiempo-Modell kombiniert die Vorzüge der Bildschirm-, Zeitachsen- und Flowchart-basierten Visualisierungsansätze mit objektorientierten Interaktionsparadigmen. Es ermöglicht eine übersichtliche Visualisierung der unterschiedlichen Aspekte einer Dokumentspezifikation und deren Manipulation. Durch Möglichkeiten zur Simulation der Auswahl von Präsentationsalternativen und zum Abfragen der vollständig konsistenten Dienstgütebereiche von Attributen erhalten Autoren eine hervorragende Unterstützung bei der Spezifikation adaptiver Dokumente.

Das für das Tiempo-Modell entwickelte Konsistenzprüfungsverfahren erlaubt eine automatische Überprüfung der Konsistenz von Dokumentspezifikationen, inklusive der enthaltenen Interaktionsmöglichkeiten. Dadurch wird eine manuelle Konsistenzprüfung überflüssig und ein Autor kann sich bei der Dokumenterstellung vollständig auf die inhaltlichen Aspekte konzentrieren.

Um eine orchestrierte Präsentation entsprechend einem Multimedia-Dokument durchführen zu können, muß ein Ablaufplan erstellt werden. Der Tiempo-Ablaufplanungsalgorithmus wählt in Abhängigkeit von durch Beobachtung gewonnenen Informationen über benötigte und verfügbare Ressourcen sowie der spezifizierten Prioritäten und Auswahlpolitiken geeignete Präsentationsalternativen aus. Er kann im Systemumgebungen, in denen eine Ressourcenzuteilung nach dem Best-Effort Prinzip oder durch Ressourcenreservierung erfolgt, eingesetzt werden. Bei einer Best-Effort Zuteilung wird immer dann, wenn sich die Menge verfügbarer Ressourcen signifikant ändert, eine Adaption des Ablaufplans berechnet und die Präsentation dementsprechend angepaßt.

Die in der Arbeit vorgestellten Konzepte wurden prototypisch als Multimedia-Dokumentensystem implementiert, das über standardisierte Protokolle mit Dokumenten-Servern kommuniziert. Dieses Dokumentensystem basiert auf einer objektorientierten Architektur, welche insbesondere die einfache Integration neuer Medien- und Darstellungsarten unterstützt. Somit kann das Tiempo-Dokumentensystem auch als Plattform zur Untersuchung weiterer Aspekte von Multimedia-Dokumenten dienen.

Das Tiempo-Modell ermöglicht auf Attributebene nur zeitlich adaptive Spezifikationen. Um zusätzliche Adaptionsoptionen zu erhalten, könnten räumlich adaptive Dokumentspezifikationen eingeführt werden, beispielsweise indem die räumliche Skalierungen von Bildern und Videos erlaubt wird. Dazu müßte das Tiempo-Modell dahingehend erweitert werden, daß räumliche Attribute, welche beispielsweise die Höhe und Breite von Medien beschreiben, flexibel durch Dienstgütebereiche spezifiziert werden können. Des weiteren müßten dann ebenfalls das vorgestellte graphische Spezifikationsverfahren, das Konsistenzprüfungsverfahren und die Ablaufplanungsalgorithmen erweitert werden.

Beim Auftreten einer Benutzerinteraktion geht die Dokumentpräsentation in einen neuen Zustand über. Um den neuen Präsentationszustand herstellen zu können, müssen die hierfür erforderlichen Dateneinheiten von Medienobjekten zur Verfügung stehen. Das vorgestellte Tiempo-Ablaufplanungsverfahren plant die Bereitstellung solcher Dateneinheiten erst ein, wenn eine Benutzerinteraktion tatsächlich aufgetreten ist. Dadurch kann der Interaktionseffekt teilweise nur mit einer gewissen Verzögerung umgesetzt werden. Um dieses Problem zu reduzieren, könnte das Ablaufplanungsverfahren dahingehend erweitert werden, daß die Aktionen zur Herstellung eines neuen Präsentationszustands so eingeplant werden, daß möglichst viele Aktionen vor dem potentiellen Auftreten der entsprechenden Benutzerinteraktion abgeschlossen sind. Dadurch würden zwar zusätzliche Ressourcen verbraucht werden, und zwar primär Speicher auf dem Präsentationsterminal und Bandbreite, die falls die Interaktionsmöglichkeiten einer Präsentation nicht genutzt werden, vergeudet sind. Aus Sicht des Benutzers und seiner Multimedia-Präsentation hätte dieser Ansatz jedoch Vorteile, da sich die Reaktionszeit des Präsentationssystems auf Benutzerinteraktionen erhöht.

Im Tiempo-Ansatz entscheidet alleine der Autor eines Multimedia-Dokuments welche Präsentationsalternativen es gibt und wie sie vom Präsentationssystem ausgewählt werden sollen. Der Nutzer eines Dokuments könnte jedoch auch in den Adaptionsprozeß miteinbezogen werden, so daß er entsprechend seinem subjektiven Empfinden entscheiden kann, wie eine Präsentation adaptiert werden soll. Beispielsweise indem der Nutzer die Möglichkeit erhält, zwischen mehreren möglichen Adaptionen zu wählen. Hierzu müßte die Ablaufplanungssteuerung im Tiempo-Präsentationssystem so erweitern werden, daß ein Nutzer über alternative Adaptionsmöglichkeiten informiert wird und eine geeignete Auswahl treffen kann.

Es gibt Ansätze Multimedia-Dokumente automatisch zu generieren. Hierbei gibt der Nutzer Suchbegriffe vor, anhand derer aus einer Datenbank passende Medien herausgesucht und nach gewissen Regeln zu einem Multimedia-Dokument zusammengestellt werden. Die hierbei angewendeten Verfahren zur Auswahl von Medien könnten mit dem Tiempo-Ablaufplanungsverfahren kombiniert werden, um nicht nur eine optimale inhaltliche Auswahl zu treffen, sondern auch die momentane Ressourcensituation zu berücksichtigen.

11 Literaturverzeichnis

[ABF97] W. Almesberger, J.-Y. L. Boudec and T. Ferrari. Scalable resource reservation for the internet. In *Proc. IEEE Conference Protocols for Multimedia Systems - Multimedia Networking (PROMSMmNet), Santiago, Chile*, 11 1997.

[AHS90] David P. Anderson, Ralf Guido Herrtwich, Carl Schaefer. SRP: A Resource Reservation Protocol for Guaranteed-Performance Communication in the Internet. Technischer Bericht: *Report No. UCB/CSD 90/562, Computer Science Division (EECS) University of California, Berkeley, CA*, 2 1990.

[AHU83] Alfread V. Aho, John E. Hopcroft, Jeffrey D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1998.

[Alle83] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Comm. ACM*, 26(11):832–843, 11 1983.

[Appl91] Computer Inc. Apple. *QuickTime Developer's Guide*. Developer technical Publications, 1991.

[Bart94] Ingo Barth. Extending the Rate Monotonic Scheduling Algorithm to Get Shorter Delays. *International Workshop on Advanced Teleservices and High-Speed Communication Architectures, Heidelberg, Germany*, S. 104–114, 9 1994.

[Bart96] Ingo Barth. Configuring Distributed Multimedia Applications using CINEMA. In *International Workshop on Multimedia Software Development, Berlin*, S. 69-76, 3 1996.

[Bart98] Ingo Barth. *Managementarchitektur für verteilte Multimedia-Anwendungen. Dissertation*, Universität Stuttgart, Fakultät Informatik, 6 1998.

[BDF+92] Ingo Barth, Gabriel Dermler, Franz Fabian, Kurt Rothermel, Frank Sembach, Robert Erfle, and Johannes Rueckert. Multimedia Document Handling - A Survey of Concepts and Methods. Technical report, IPVR - IBM, 9 1992.

[BDH+93] Ingo Barth, Gabriel Dermler, Tobias Helbig, Kurt Rothermel, Frank Sembach, Thomas Wahl. CINEMA: Eine konfigurierbare, integrierte Multimedia-Architektur. In *Verteilte Multimedia Systeme*, S. 33–47. Wolfgang Effelsberg, Kurt Rothermel, 2 1993.

[BFF+96] L. Berc, B. fenner, R. Frederick and S. McCanne. *RTP payload format for JPEG-compressed video*. Proposed Standard 2035, Internet Engineering Task Force, 10 1996.

[BHL91] G. Blakowski, J. Hübel, U. Langrehr. Tools for Specifying and Executing Synchronized Multimedia Presentations. In Proc. *2nd International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg*, 11 1991.

[Boga92] R. Bogaschewsky. Hypertext-/Hypermedia-Systeme - Ein Überblick. *Informatik-Spektrum* 15(3): 127-143, 1992.

[Bole95] Dietrich Boles. Das IMRA-Modell - Modellierung interaktiver multimedialer Präsentationen. In *Proc. Hypertext - Information Retrieval - Multimedia (HIM'95), Konstanz*, S. 61-75, 1995.

[BPS98] Tim Bray, Jean Paoli, C.M. Sperberg-McQueen. *Extensible Markup Language (XML) 1.0*, W3C Recommendation 19980210, 2 1998.

[Bruc72] B.C. Bruce. A Model for Temporal Reference and its Applicants in a Question Answering Program. *Artificial Intelligence*, 3:1–12, 1972.

[Bult93] Dick C.A. Bulterman. Specification and Support of Adaptable Networked Multimedia. In *ACM Multimedia Systems*, Springer-Verlag, 1993.

[BuZe92] M. Cecelia Buchanan and Polle T. Zellweger. Scheduling Multimedia Documents Using Temporal Constraints. In *Proc. of the 3rd International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, USA*, S. 223–235, 1992.

[BuZe93] M. Cecilia Buchanan and Polle T. Zellweger. Automatic Temporal Layout Mechanisms. *In Proc. of ACM 1st Intl. Conference on Multimedia, Anaheim, USA*, S. 341 – 350, 1993.

[CaHa74] R. H. Campbell and A. N. Habermann. The Specification of Process Synchronisation by Path Expressions. In G. Goos and J. Hartmanis, editors, *Lecture notes in Computer Science No. 16, Operating Systems*, S. 89–102. Springer Verlag, 1974.

[CoRo83] James E. Coolahan and Nicholas Roussopoulos. Timing Requirements for Time-Driven systems Using Augmented Petri Nets. In *IEEE Transactions and Software Engineering*, Vol. SE-9, No. 5, September 1983.

[CPS96] 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*, S. 329-339, 1996.

[DMP89] Dechter, Meiri, Pearl. Temporal Constraint Networks. In Proc. *First International Conference on Principles of Knowledge Representation and Reasoning*, S. 83-93, 5 1989.

[Drap93] George D. Drapeau. Synchronization in the MAEstro Multimedia Authoring Environment. In *Proc. of ACM 1st Intl. Conference on Multimedia, Anaheim, USA*, S. 331–340, 1993.

[EiHo93] H. Eirund, M. Hofmann. Designing Multimedia Presentations. In *Tagungsband GI-Fachtagung für Hypermedia*, Zürich, Reihe: Informatik Aktuell. Springer-Verlag, 1993.

[EHV93] J. L. Encarnacao, W. Hübner, K. Väänänen. Autorenwerkzeuge für multimediale Informationssysteme. In *Informationstechnik und Technische Informatik*, 35(2):31-38, 1993.

[ENK+94] S. Eun, E. S. No, H. C. Kim, H. Yoon, S. R. Maeng. Eventor: an authoring system for interactive multimedia presentations. In *Multimedia Systems*, Vol 2, No 2, S. 129-140, 1994.

[FGM+97] R. Fielding, J. Gettys, J. Mogul, H. Nielsen, T. Berners-Lee. *Hypertext transfer protocol – HTTP/1.1*, RFC 2068, Internet Engineering Task Force, 1 1997.

[Fisc95] Steffen Fischer. Autorenunterstützende Konsistenzprüfung multimedialer Dokumente. Diplomarbeit Nr. 1341, IPVR, Universität Stuttgart, 4 1995.

[Flan99] David Flanagan. *Java in a Nutshell, 3rd Edition*. O'Reilly & Associates, Inc., 1999.

[Fowl97] Martin Fowler. *Analysis Patterns - Reusable Object Models*. Addison-Wesley, 1997.

[Fran95] Georg Franke. Multimedia-Authoring im Tiempo-Projekt. Diplomarbeit Nr. 1251, IPVR, Universität Stuttgart, 6 1995.

[Gall91] Didier Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. In *Communications of the ACM*, 34(4):46–58, 4 1991.

[Gesc97] Jan Gecsei. Adaptation in Distributed Multimedia Systems. In *IEEE Multimedia*, S. 58-66, April-Juni 1997.

[Gibb91] Simon Gibbs. Composite Multimedia and Active Objects. *OOPSLA '91, Phoenix*, S. 97–112, 10 1991.

[GKK+92] K. Gottheil, H.-J. Kaufmann, Th. Kern, R. Zhao. *X und Motif*. Springer-Verlag, 1992.

[Good90] D. Goodman. *The Complete HyperCard 2.0 Handbook*. Bantam Books, Inc., 1990.

[GRJ99] G. Booch, J. Rumbaugh, I. Jacobson. *Unified Modeling Language User Guide*. Addison-Wesley, 1999.

[HaRo00] Jürgen Hauser, Kurt Rothermel. Specification and Implementation of an Extensible Multimedia System. In *Proc. Interactive Distributed Multimedia Systems and Telecommunication Services; IDMS'2000, Enschede, Holland*, Oktober 2000.

[Helb96] Tobias Helbig. *Kommunikation und Synchronisation multimedialer Datenströme in verteilten Systemen*. Dissertation, Universität Stuttgart, Fakultät Informatik, 1996.

[HFG+98] D. Hoffman, G. Fernando, V. Goyal und M. Civanlar. *RTP payload format for MPEG1/MPEG2 video*. Proposed Standard 2250, Internet Engineering Task Force, 1 1998.

[Hoar85] C.A.R Hoare. *Communicating Sequential Processes*. Engelwood Cliffs NJ: Prentice-Hall, 1985.

[Hoep91] P. Hoepner. Synchronisation der Praesentation von Multimediaobjekten - Modell und Beispiele. In J. Encarnacao, editor, *Informatik-Fachberichte 293, Telekommunikation und multimediale Anwendungen der Informatik*, S. 455–464. GI, Springer-Verlag, 10 1991.

[Hosc98] Philipp Hoschka (Ed.). *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification*, W3C Proposed Recommendation, 4 1998.

[HSA89] M. E. Hodges, R. M. Sasnett, M. S. Ackerman. A Construction Set for Multimedia Applications. In *IEEE Software*, S. 37-43, 1 1989.

[HSR92] Rei Hamakawa, Hidekazu Sakagami, and Jun Rekimoto. Audio and Video Extensions to Graphical User Interface Toolkits. In *Proc. of the Third International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, USA,* S. 356-361, 11 1992.

[HSR94] Rei Hamakwa, Hidekazu Sakagami, and Jun Rekimoto. Mbuild - Multimedia Data Builder with Box and Glue. In *Proc. of IEEE 1st Intl. Conference on Multimedia Computing and Systems, Boston, USA*, S. 520–525, 1994.

[HyTi92] HyTime. *Information technology - Hypermedia/Time-based Structuring Language (HyTime)*. ISO/IEC DIS 10744, 1992.

[Kale93] Kaleida. *Kaleida, ScriptX, and the Multimedia Market.* Kaleida Labs Inc., 1945 Charstom Road, Mountain View, USA CA94043, 1993.

[KeDu96] Cherif Keramane and Andrzej Duda. Interval Expressions - a Functional Model for Interactive Dynamic Multimedia Presentations. In Proc. of IEEE Multimedia, S. 283-286, 1996.

[KeLo91] Somnuk Keretho and Rasiah Logananthara j. Qualitative and Quantitative Time Interval Constraint Networks. In *Proc. of ACM, San Antonio*, 1991.

[KiSo95] Michelle Y. Kim and Junehwa Song. Multimedia Documents with Elastic Time. In Proc. ACM Multimedia, San Fransisco, CA USA, S. 143- 154, 1995.

[KnPl84] D. Knuth and M. Plass. Breaking Paragraphs into Lines. In *Software-Practice and Experience*, Vol. 11, S. 1119-1184, 1984.

[KrCa92] Francis Kretz and Francoise Calaitis. Standardizing Hypermedia Information Objects. In *IEEE Communications Magazine*, 5 1992.

[LaKe95] Layaida, Kermane. Maintaining Temporal Consistency of Multimedia Documents. In *Proc. ACM Workshop on Effective Abstractions in Multimedia*, 11 1995.

[LDK94] Guiseppe Lo Re, Luca Delgrossi, Thomas Kober. TCP/IP Protocols over ATM Networks: Experiments and Results. Technischer Bericht *TR 43.9413, IBM ENC*, 1994.

[LiGh90] T.D.C. Little and A. Ghafor. Synchronization and Storage Models for Multimedia Objects. In *IEEE Journal on Selected Areas in Communication*, 8(3):413-427, 1990.

[LiKo92] T. D. C. Little and F. Koa. An Intermedia Skew Control System for Multimedia Data Presentation. In *3rd Intl. Workshop on Network and Operating Systems Support for Digital Audio and Video*, S. 121 – 132, 11 1992.

[Liou91] Ming Liou. Overview of the px64 kbit/s Video Coding Standard. In *Communications of the ACM*, 34(4):59–63, 4 1991.

[Lütj94] O. Lütjens. Neue Schale, neuer Kern. In *Screen Multimedia*. S. 60-63, 6 1994.

[Maye95] Markus Mayer. Entwicklung eines Zugriffsystems für multimediale Objekte. Studienarbeit Nr. 1439, IPVR, Universität Stuttgart, 6 1995.

[MHEG94] MHEG. *Information Technology - Coded Representation of Multimedia and Hypermedia Information Objects*. ISO/IEC IS 13522.

[Mino86] M. Minoux. *Mathematical Programming - Theory and Algorithms*, John Wiley and Sons. 1986.

[NaSm95] Klara Nahrstedt, Jonathan M. Smith. The QOS Broker. In *IEEE Multimedia*, 2(1):53–67, 1995.

[Neum75] Klaus Neumann. *Operations Research Verfahren, Band I*, Carl Hanser Verlag, 1975.

[PAP95] D. Papadias, Y. Theodoridis. Spatial relations, Minimum Bounding Rectangles, and Spatial Data Structures. In International Journal of Geographic Information Systems, 1995.

[PeLi96] Maria Jose Perez-Luque and Thomas D. C. Little. A Temporal Reference Framework for Multimedia Synchronization. In *IEEE Journal on Selected Areas in Communication*, 14(1), 1 1996.

[PrRa93] B. Prabhakaran and S. V. Raghavan. Synchronization Models for Multimedia Presentation with User Participation. In *Proc. of ACM 1st Intl. Conference on Multimedia, Anaheim, USA*, S. 157-166, 1993.

[QWG93] Naveed U. Qazi, Miae Woo, Arif Ghafoor. A Synchronization and Communication Model for Distributed Multimedia Objects. In *Proc. of ACM 1st Intl. Conference on Multimedia, Anaheim, USA*, S. 147-155, 1993.

[RJM+93] Guido van Rossum, Jack Jansen, K. Sjoerd Mullender and Dick C.A. Bulterman. CMIFed: A Presentation Environment for Portable Hypermedia Documents. In *Proc. ACM Multimedia*, S. 183-188, 1993.

[RoHe95] Kurt Rothermel, Tobias Helbig. An Adaptive Stream Synchronization Protocol. In *Proc. 5th International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, New Hampshire, USA*, S. 189–202, 4 1995.

[ROH+97] Lloyd Rutledge, Jacco van Ossengruggen, Lynda Hardman and Dick C.A. Bultermann. A Framework for Generating Adaptable Hypermedia Documents. In *Proc. of ACM Multimedia 97, Seattle WA, USA*, 11 1997.

[SCF+96] H. Schulzrinne, S. Casner, R. Frederick und V. Jacobson, *RTP: A transport protocol for real-time applications*. Proposed Standard 1889, Internet Engineering Task Force, 1 1996

[Schu95] H. Schulzrinne. Internet Services: from electronic mail to real-time multimedia. In *Proc. Kommunikation in Verteilten Systemen (KIVS'95), Chemnitz*, S. 21-34, 2 1995.

[SDL+96] P. Senac, Michel Diaz, Alain Leger, and Pierre de Saqui-Sannes. Modeling Logical and Temporal Synchronization in Hypermedia Systems. In *IEEE Journal on Selected Areas in Communications*, 14(1):84-104, 1996

[Sedg88] Robert Sedgewick. *Algorithms*, Second Edition, Addison-Wesley, S. 476-478, 1999.

[SKD96] J. Schnepf, J. A. Konstan, and D. H.-C. Du. Doing FLIPS: FLexible Interactive Presentation Synchronization. *In IEEE Jounal on Selected Areas in Communications*, 14(1):114-125, 1996.

[SLF+96] Timothy K. Shih, Steven K. C. Lo, Szu-Jan Fu, and Julian B. Chang. Using Interval Temporal Logic and Inference Rules fo rthe Automatic Generation of Multimedia Presentations. In Proc. of IEEE Multimedia, S. 425-428, 1996.

[SRL97] H. Schulzrinne, A. Rao und R. Lamphier. *Real time streaming protocol (RTSP)*, Internet Draft, Internet Engineering Task Force, 3 1997.

[SRR90] R. Steinmetz, Johannes Rückert, W. Racke. Multimedia-Systeme. In *Informatik Spektrum*, 13(5):280–282, 1990.

[Stei90] Ralf Steinmetz. Synchronisation Properties in Multimedia Systems. In *IEEE Journal on Selected Areas in Communications*, 8(3):401–412, 4 1990.

[Stei92] Ralf Steinmetz. Multimedia Synchronisation Techniques: Experiences Based on Different System Structures. In *Multimedia Communications, Monterey, USA CA*, 4th IEEE COMSOC International Workshop, 4 1992.

[Stei93a] B. Steinbrink. Multimedia-Regisseure - Autorensysteme und -sprachen im Vergleich, *c't*, 1 1993.

[Stei93b] Ralf Steinmetz. *Multimedia Technologie*. Springer Verlag, 1993.

[Stei94] Ralf Steinmetz. Multimedia Operating Systems: Resource Reservation, Scheduling, File Systems, and Architecture. Technischer Bericht *TR 43.94062, IBM European Networking Center Heidelberg*, 1994.

[StEn93] Ralf Steinmetz, Clemens Engler. Human Perception of Media Synchronization. *Technical Report 43.9310, IBM ENC, Heidelberg*, 1993.

[Stro87] Bjarne Stroustrup. *The C++ programming language*. Addison-Wesley, 1987.

[TaHo94] Wassim Tawbi, Eric Horlait. Expression and Management of QoS in Multimedia Communication Systems. *Annals of Telecommunications*, T.49(5-6):282–296, 5-6 1994.

[Tane88] Andrew S. Tanenbaum. *Computer networks*. Prentice-Hall, Englewood Cliffs, NJ, 2nd ed. edition, 1988. XV, 658 S.

[ThKl96] H. Thimm and W. Klas. delta-sets for optimized reactive adaptive playout management in distributed multimedia database systems. In Proc. 12th IEEE Int. Conf. on Data Engineering, New Orleans, USA, S. 584-592, 1996.

[Topo90] C. Topolcic. Experimental Internet Stream Protocol, Version 2 (ST-II). *RFC 1190*, 10 1990.

[Vään92] K. Väänänen. Interfaces to Hypermedia: Communicating the Structure and Interaction Possibilities to the Users. In *Proc. of the 2nd Eurographics Workshop on Multimedia, Darmstadt*, 5 1992.

[vBee92] Peter van Beek. Reasoning about qualitative temporal information. *Artificial Intelligence*, 8(956):297 – 326, 12 1992.

[ViKa86] M. Vilain and H.A. Kautz. Constraint propagation algorithms for temporal reasoning. In *AAAI-86 Philadelphia, PA*, S. 132 – 144, 1986.

[Volg94] T. J. Volgger. Trendwende. *Screen Multimedia*, S. 50-53, 3 1994.

[WaRo94] Thomas Wahl and Kurt Rothermel. Representing Time in Multimedia Systems. In *Proc. of IEEE 1st Intl. Conference on Multimedia Computing and Systems, Boston, USA*, S. 538–543, 1994.

[West91] N. West. Multimedia Design Tools. *MACWORLD*, S. 194-201, 11 1991.

[Wira97] Stefan Wirag. Modeling of Adaptable Multimedia Documents. In *Proc. Interactive Distributed Multimedia Systems and Telecommunication Services; 4th International Workshop, IDMS'97, Darmstadt*, S. 420-429, 9 1997.

[Wira99a] Stefan Wirag. Specification and Scheduling of Adaptive Multimedia Documents. *Fakultätsbericht 1999/4, Universität Stuttgart*, 1 1999.

[Wira99b] Stefan Wirag. Adaptive Scheduling of Multimedia Documents, In *Proc. GI/ITG Kommunikation in Verteilten Systemen, Darmstadt*, S. 450-461, 3 1999

[Wira99c] Stefan Wirag. Scheduling of Adaptive Multimedia Documents. In *Proc. of IEEE Intl. Conference on Multimedia Computing and Systems, Florenz, Italien*, 6 1999.

[WiRo98] Stefan Wirag, Kurt Rothermel. Adaptive Multimedia Documents, *Journal of Computing and Information Technology (CIT)*. 6(3), 9 1998.

[WRW95] Stefan Wirag, Kurt Rothermel and Thomas Wahl. Modeling Interaction with HyTime. In *Proc. GI/ITG Kommunikation in Verteilten Systemen, Germany, Chemnitz*, S. 188-202, 2 1995

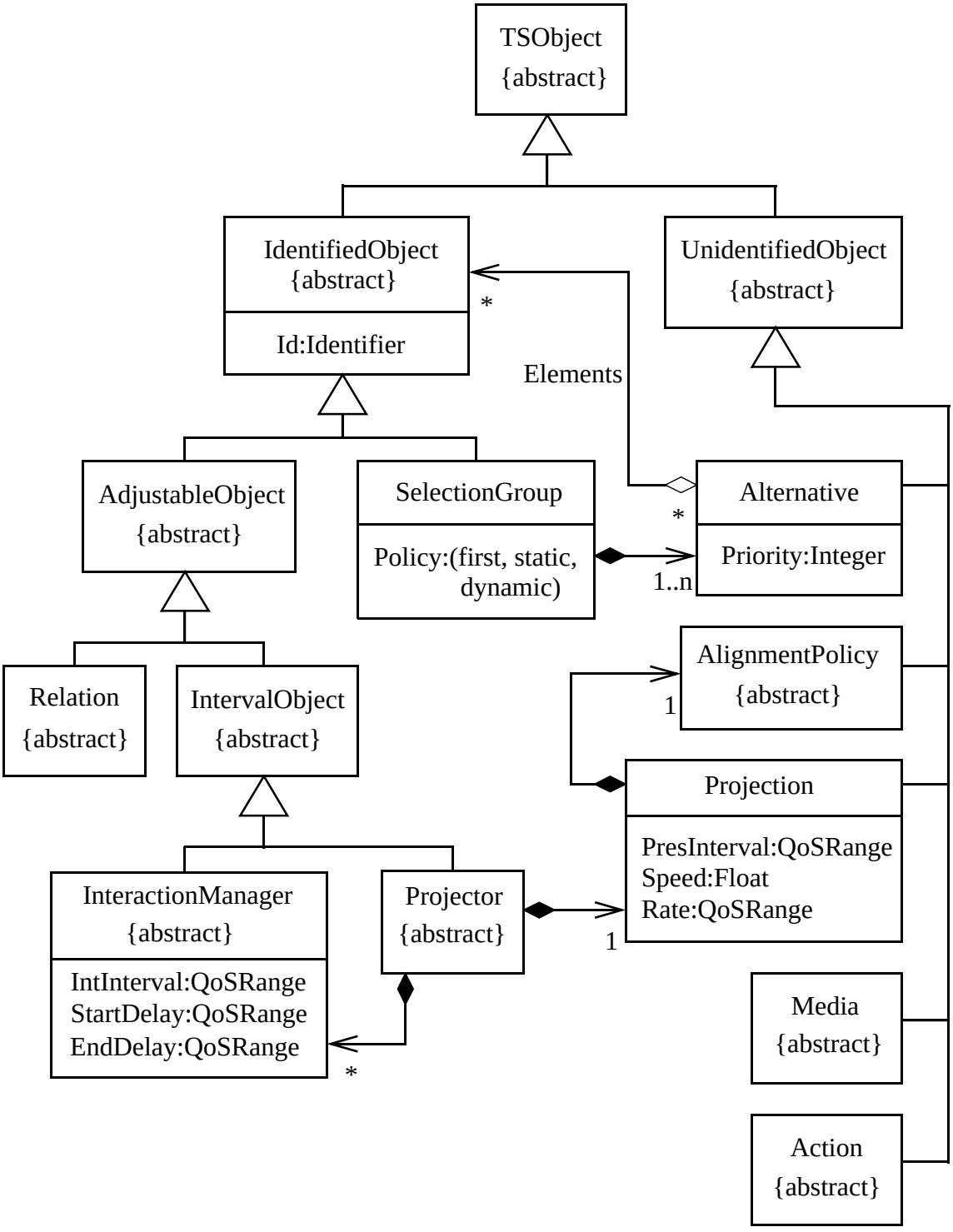
[WWR95] Thomas Wahl, Stefan Wirag and Kurt Rothermel. TIEMPO: Temporal Modeling and Authoring of Interactive Multimedia. In *Proc. of IEEE 2nd Intl. Conference on Multimedia Computing and Systems, Washington DC*, S. 274-277, 5 1995

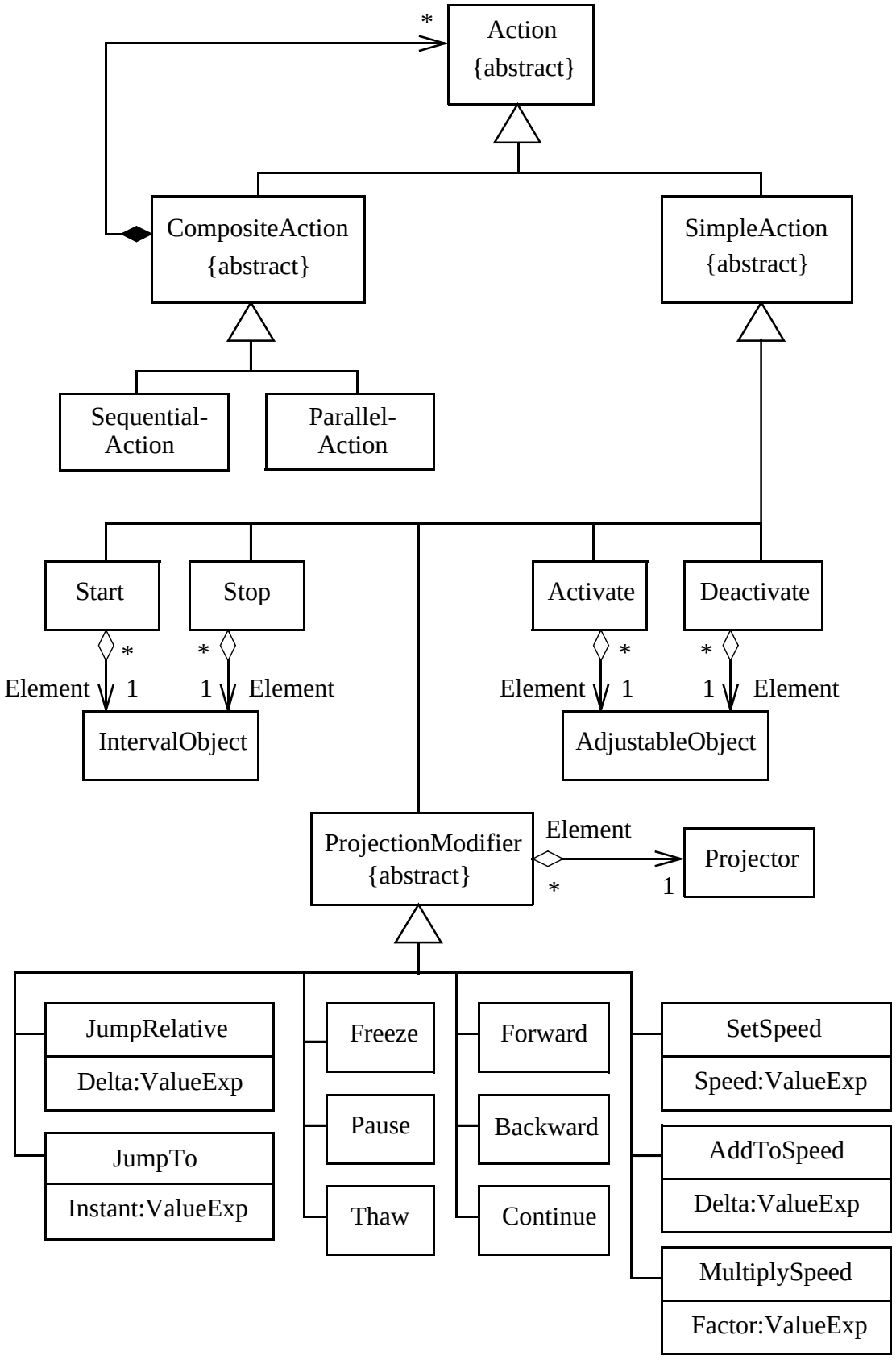
[ZDE+93] Lixia Zhang, Steve Deering, Debora Estrin, Scott Shanker, Daniel Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, 9 1993.

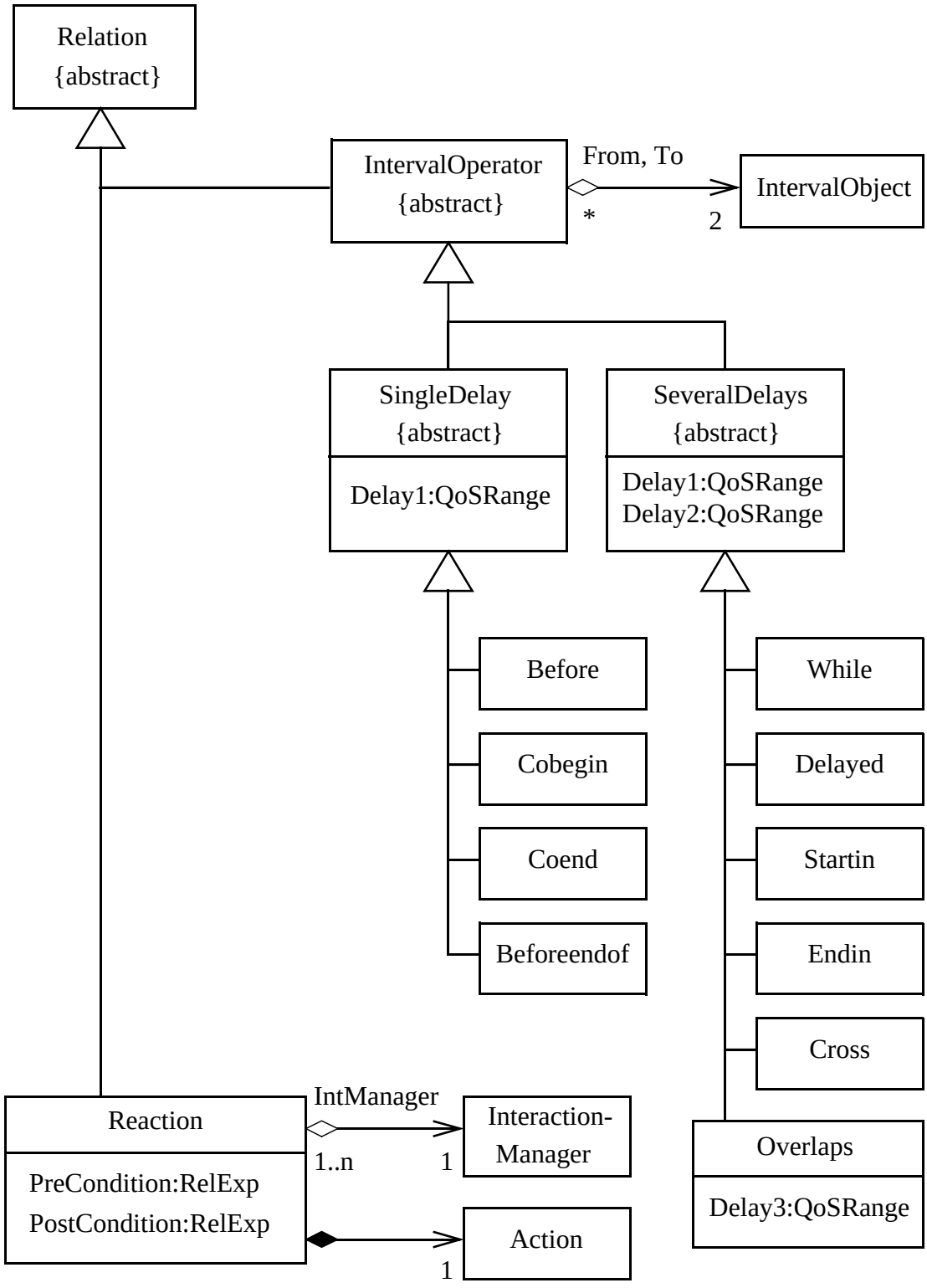
[Zhu97] C. Zhu. *RTP payload format for H.263 video streams*. Proposed Standard 2190, Internet Engineering Task Force, 10 1996.

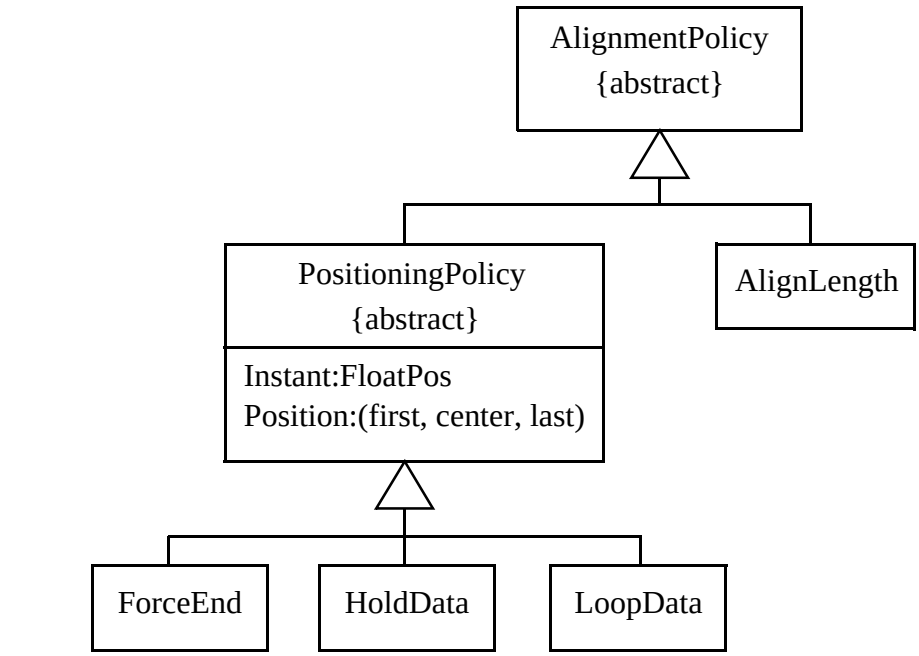
Anhang A. Tiempo-Dokumentensprachmodell

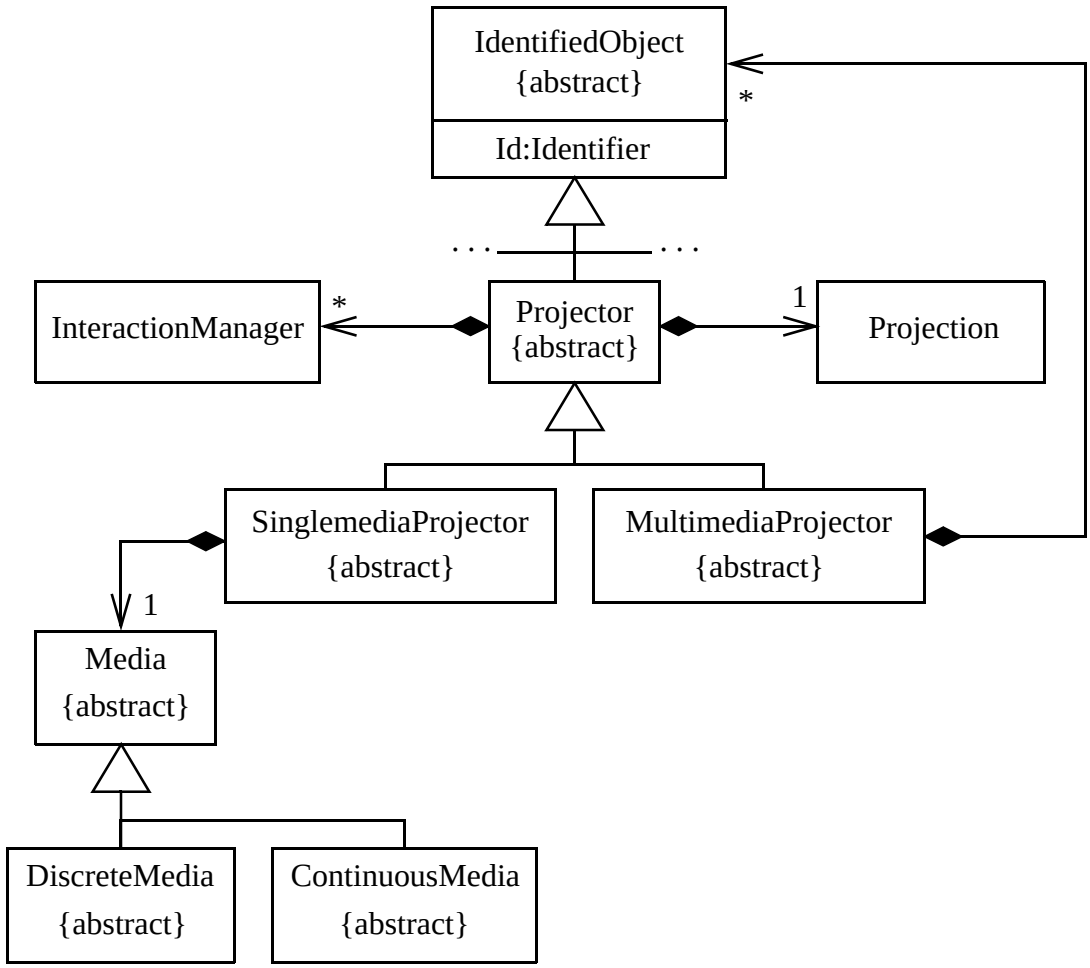
A.1 Die Tiempo-Dokumentenspracharchitektur in UML-Notation




Action
{abstract}
*
CompositeAction
{abstract}
SimpleAction
{abstract}
Sequential-
Action
Parallel-
Action
Start
Stop
Activate
Deactivate
Element
1
*
IntervalObject
AdjustableObject
ProjectionModifier
{abstract}
Projector
JumpRelative
Delta:ValueExp
JumpTo
Instant:ValueExp
Freeze
Pause
Thaw
Forward
Backward
Continue
SetSpeed
Speed:ValueExp
AddToSpeed
Delta:ValueExp
MultiplySpeed
Factor:ValueExp


Relation
{abstract}
IntervalOperator
{abstract}
From, To
*
2
IntervalObject
SingleDelay
{abstract}
Delay1:QoSRange
SeveralDelays
{abstract}
Delay1:QoSRange
Delay2:QoSRange
Before
Cobegin
Coend
Beforeendof
While
Delayed
Startin
Endin
Cross
Reaction
PreCondition:RelExp
PostCondition:RelExp
IntManager
1..n
1
Interaction-
Manager
1
Action
Overlaps
Delay3:QoSRange


AlignmentPolicy
{abstract}
PositioningPolicy
{abstract}
Instant:FloatPos
Position:(first, center, last)
AlignLength
ForceEnd
HoldData
LoopData


IdentifiedObject
{abstract}
Id:Identifier
*
InteractionManager
*
Projector
{abstract}
1
Projection
SinglemediaProjector
{abstract}
MultimediaProjector
{abstract}
1
Media
{abstract}
DiscreteMedia
{abstract}
ContinuousMedia
{abstract}

Attributtypen:

Identifier: Eindeutiger Name im Dokument

Integer: Integerkonstante

FloatPos: Gleitkommakonstante

Float: Ist ein Float-Ausdruck

Float-Ausdruck ⟶
Gleitkommakonstante
| '+' Gleitkommakonstante
| '-' Gleitkommakonstante

QoSRange: Ist ein QoS-Ausdruck

QoS-Ausdruck ⟶
Wert
| '[' Wertepaare ']'
| '[' Wertepaare ']:' Politik

Wertepaare ⟶
PriorisierterWert
| Wertepaare ',' PriorisierterWert

PriorisierterWert ⟶
Wert
| Wert ':' Integerkonstante

Wert ⟶
'?'
| Integerkonstante
| Gleitkommakonstante

Politik ⟶
'f'
| 's'
| 'd'

RelExpr: Ist ein ODER-Ausdruck

ODER-Ausdruck ⟶
UND-Ausdruck
| ODER-Ausdruck '|' UND-Ausdruck

UND-Ausdruck ⟶
Gleichheitsausdruck
| UND-Ausdruck '&' Gleichheitsausdruck

Gleichheitsausdruck ⟶
    Relationaler-Ausdruck
    | Gleichheitsausdruck '=' Relationaler-Ausdruck
    | Gleichheitsausdruck '<>' Relationaler-Ausdruck

Relationaler-Ausdruck ⟶
    Additiver-Ausdruck
    | Relationaler-Ausdruck '<' Additiver-Ausdruck
    | Relationaler-Ausdruck '>' Additiver-Ausdruck
    | Relationaler-Ausdruck '<=' Additiver-Ausdruck
    | Relationaler-Ausdruck '>=' Additiver-Ausdruck

ValueExpr: Ist ein Additiver-Ausdruck

Additiver-Ausdruck ⟶
    Multiplikativer-Ausdruck
    | Additiver-Ausdruck '+' Multiplikativer-Ausdruck
    | Additiver-Ausdruck '-' Multiplikativer-Ausdruck

Multiplikativer-Ausdruck ⟶
    Unärer-Ausdruck
    | Multiplikativer-Ausdruck '*' Unärer-Ausdruck
    | Multiplikativer-Ausdruck '/' Unärer-Ausdruck
    | Multiplikativer-Ausdruck '%' Unärer-Ausdruck

Unärer-Ausdruck ⟶
    Primärer-Ausdruck
    | '+' Primärer-Ausdruck
    | '-' Primärer-Ausdruck
    | '!' Primärer-Ausdruck

Primärer-Ausdruck ⟶
    '(' ODER-Ausdruck ')'
    | Integerkonstante
    | Gleitkommakonstante
    | String-Konstante
    | State-Attributname eines Interaktionsmanagers

A.2 XML-DTD eines Tiempo-Dialekts

```
<!-- Multimedia-Projektoren -->

<!ELEMENT  Document  (Projection, Window+, (Before | Cobegin | Coend | Beforeendof | While |
                        Delayed | Startin | Endin | Cross | Overlaps)*, Reaction*, SelectionGroup*) >
<!ATTLIST Document
          Tiempo          NMTOKEN       #FIXED  "MultimediaProjector"
          Id              ID            #REQUIRED
>

<!ELEMENT  Window  (Projection, (VideoPro | PicturePro | LongTextPro | AudioPro | ShortTextPro |
                        Composite | ButtonPro | SliderPro)+, (Before | Cobegin | Coend |
                        Beforeendof | While | Delayed | Startin | Endin | Cross | Overlaps)*,
                        Reaction*, SelectionGroup*) >
<!ATTLIST  Window
          Tiempo          NMTOKEN       #FIXED  "MultimediaProjector"
          Id              ID            #REQUIRED
          Width           CDATA         "300"
          Height          CDATA         "300"
          XPos            CDATA         "0"
          YPos            CDATA         "0"
          ZPos            CDATA         "0"
          Background      CDATA         "grey"
>

<!ELEMENT  Composite  (Projection, (VideoPro | PicturePro | LongTextPro | AudioPro | ShortTextPro |
                        Composite | ButtonPro | SliderPro)+, (Before | Cobegin | Coend |
                        Beforeendof | While | Delayed | Startin | Endin | Cross | Overlaps)*,
                        Reaction*, SelectionGroup*) >
<!ATTLIST Composite
          Tiempo          NMTOKEN       #FIXED  "MultimediaProjector"
          Id              ID            #REQUIRED
>


<!-- Einfache Projektoren -->

<!ELEMENT  ShortTextPro  (Projection, ShortText) >
<!ATTLIST  ShortTextPro
          Tiempo     NMTOKEN                      #FIXED  "SinglemediaProjector"
          Id         ID                           #REQUIRED
          XPos       CDATA                        "0"
          YPos       CDATA                        "0"
          ZPos       CDATA                        "0"
          Font       (times | helvetica)          "times"
          Size       (S1 | S2 | S3 | S4 | S5 | S6 | S7)    "S6"
```

```
        Style       (normal | italic | bold | boldItalic)     "normal"
        Color       CDATA                                     "black"
>

<!ELEMENT  LongTextPro  (Projection, LongText) >
<!ATTLIST  LongTextPro
        Tiempo      NMTOKEN     #FIXED  "SinglemediaProjector"
        Id          ID          #REQUIRED
        XPos        CDATA       "0"
        YPos        CDATA       "0"
        ZPos        CDATA       "0"
        Width       CDATA       "200"
        Height      CDATA       "400"
>

<!ELEMENT  PicturePro  (Projection, Picture) >
<!ATTLIST  PicturePro
        Tiempo      NMTOKEN     #FIXED  "SinglemediaProjector"
        Id          ID          #REQUIRED
        XPos        CDATA       "0"
        YPos        CDATA       "0"
        ZPos        CDATA       "0"
>

<!ELEMENT  AudioPro  (Projection, Audio) >
<!ATTLIST  AudioPro
        Tiempo      NMTOKEN     #FIXED  "SinglemediaProjector"
        Id          ID          #REQUIRED
        Volume      CDATA       "50"
        Balance     CDATA       "50"
>

<!ELEMENT  VideoPro  (Projection, Video) >
<!ATTLIST  VideoPro
        Tiempo      NMTOKEN     #FIXED  "SinglemediaProjector"
        Id          ID          #REQUIRED
        XPos        CDATA       "0"
        YPos        CDATA       "0"
        ZPos        CDATA       "0"
>

<!ELEMENT  ButtonPro  (Projection, ButtonIntMng, (ShortText | Picture) ) >
<!ATTLIST  ButtonPro
        Tiempo      NMTOKEN     #FIXED  "SinglemediaProjector"
        Id          ID          #REQUIRED
        XPos        CDATA       "0"
        YPos        CDATA       "0"
        ZPos        CDATA       "0"
```

```
        Color       CDATA       "navy"
>

<!ELEMENT SliderPro (Projection, SliderIntMng, ShortText?) >
<!ATTLIST SliderPro
        Tiempo      NMTOKEN     #FIXED "SinglemediaProjector"
        Id          ID          #REQUIRED
        XPos        CDATA       "0"
        YPos        CDATA       "0"
        ZPos        CDATA       "0"
        Width       CDATA       "100"
        Color       CDATA       "navy"
        Minimum     CDATA       "0"
        Maximum     CDATA       "100"
>


<!-- Interaktionsmanager -->

<!ELEMENT ButtonIntMng EMPTY>
<!ATTLIST ButtonIntMng
        Tiempo      NMTOKEN             #FIXED "InteractionManager"
        Id          ID                  #REQUIRED
        IntInterval CDATA               "[0:0,?:0]:d"
        StartDelay  CDATA               "[0:0,?:0]:d"
        EndDelay    CDATA               "[0:0,?:0]:d"
        State       (pressed | released)  "released"
>

<!ELEMENT SliderIntMng EMPTY>
<!ATTLIST SliderIntMng
        Tiempo      NMTOKEN     #FIXED "InteractionManager"
        Id          ID          #REQUIRED
        IntInterval CDATA       "[0:0,?:0]:d"
        StartDelay  CDATA       "[0:0,?:0]:d"
        EndDelay    CDATA       "[0:0,?:0]:d"
        State       CDATA       "0"
>


<!-- Medienelemente -->

<!ELEMENT ShortText EMPTY >
<!ATTLIST ShortText
        Tiempo      NMTOKEN     #FIXED "DiscreteMedia"
        Data        CDATA       #REQUIRED
>
```

```
<!ELEMENT LongText EMPTY >
<!ATTLIST LongText
        Tiempo      NMTOKEN         #FIXED "DiscreteMedia"
        Coding      (ASCII | HTML)  #REQUIRED
        Size        CDATA           #REQUIRED
        Data        CDATA           #REQUIRED
>

<!ELEMENT Picture EMPTY >
<!ATTLIST Picture
        Tiempo      NMTOKEN             #FIXED "DiscreteMedia"
        Coding      (GIF | JPEG | XPM)  #REQUIRED
        Width       CDATA               #REQUIRED
        Height      CDATA               #REQUIRED
        Size        CDATA               #REQUIRED
        Data        CDATA               #REQUIRED
>

<!ELEMENT Audio EMPTY >
<!ATTLIST Audio
        Tiempo      NMTOKEN         #FIXED "ContinuousMedia"
        Coding      (WAV | MP3)     #REQUIRED
        Length      CDATA           #REQUIRED
        Rate        CDATA           #REQUIRED
        UnitSize    CDATA           #REQUIRED
        Data        CDATA           #REQUIRED
>

<!ELEMENT Video EMPTY >
<!ATTLIST Video
        Tiempo      NMTOKEN         #FIXED "ContinuousMedia"
        Coding      (MPEG | MJPEG)  #REQUIRED
        Width       CDATA           #REQUIRED
        Height      CDATA           #REQUIRED
        Length      CDATA           #REQUIRED
        Rate        CDATA           #REQUIRED
        UnitSize    CDATA           #REQUIRED
        Data        CDATA           #REQUIRED
>


<!-- Projektion -->

<!ELEMENT Projection (AlignLength | ForceEnd | LoopData | HoldData) >
<!ATTLIST Projection
        Tiempo      NMTOKEN     #FIXED "Projection"
        PresInterval CDATA      "[0:0,?:0]:d"
        Speed       CDATA       "1.0"
```

```
        Rate        CDATA           "1.0"
>

<!ELEMENT AlignLength EMPTY >
<!ATTLIST AlignLength
        Tiempo      NMTOKEN     #FIXED "AlignmentPolicy"
>

<!ELEMENT ForceEnd EMPTY >
<!ATTLIST ForceEnd
        Tiempo      NMTOKEN         #FIXED "AlignmentPolicy"
        Instant     CDATA           "0"
        Position    (first | center | last) "first"
>

<!ELEMENT LoopData EMPTY >
<!ATTLIST LoopData
        Tiempo      NMTOKEN         #FIXED "AlignmentPolicy"
        Instant     CDATA           "0"
        Position    (first | center | last) "first"
>

<!ELEMENT HoldData EMPTY >
<!ATTLIST HoldData
        Tiempo      NMTOKEN         #FIXED "AlignmentPolicy"
        Instant     CDATA           "0"
        Position    (first | center | last) "first"
>


<!-- Intervalloperatoren -->

<!ELEMENT Before EMPTY >
<!ATTLIST Before
        Tiempo      NMTOKEN     #FIXED "IntervalOperator"
        Id          ID          #REQUIRED
        From        IDREF       #REQUIRED
        To          IDREF       #REQUIRED
        Delay1      CDATA       "[0:0,?:0]:d"
>

<!ELEMENT Beforeendof EMPTY >
<!ATTLIST Beforeendof
        Tiempo      NMTOKEN     #FIXED "IntervalOperator"
        Id          ID          #REQUIRED
        From        IDREF       #REQUIRED
        To          IDREF       #REQUIRED
```

```
        Delay1      CDATA       "[0:0,?:0]:d"
>

<!ELEMENT Cobegin EMPTY >
<!ATTLIST Cobegin
        Tiempo      NMTOKEN     #FIXED "IntervalOperator"
        Id          ID          #REQUIRED
        From        IDREF       #REQUIRED
        To          IDREF       #REQUIRED
        Delay1      CDATA       "[0:0,?:0]:d"
>

<!ELEMENT Coend EMPTY >
<!ATTLIST Coend
        Tiempo      NMTOKEN     #FIXED "IntervalOperator"
        Id          ID          #REQUIRED
        From        IDREF       #REQUIRED
        To          IDREF       #REQUIRED
        Delay1      CDATA       "[0:0,?:0]:d"
>

<!ELEMENT While EMPTY >
<!ATTLIST While
        Tiempo      NMTOKEN     #FIXED "IntervalOperator"
        Id          ID          #REQUIRED
        From        IDREF       #REQUIRED
        To          IDREF       #REQUIRED
        Delay1      CDATA       "[0:0,?:0]:d"
        Delay2      CDATA       "[0:0,?:0]:d"
>

<!ELEMENT Delayed EMPTY >
<!ATTLIST Delayed
        Tiempo      NMTOKEN     #FIXED "IntervalOperator"
        Id          ID          #REQUIRED
        From        IDREF       #REQUIRED
        To          IDREF       #REQUIRED
        Delay1      CDATA       "[0:0,?:0]:d"
        Delay2      CDATA       "[0:0,?:0]:d"
>

<!ELEMENT Endin EMPTY >
<!ATTLIST Endin
        Tiempo      NMTOKEN     #FIXED "IntervalOperator"
        Id          ID          #REQUIRED
        From        IDREF       #REQUIRED
        To          IDREF       #REQUIRED
        Delay1      CDATA       "[0:0,?:0]:d"
```

```
        Delay2      CDATA       "[0:0,?:0]:d"
>

<!ELEMENT  Cross  EMPTY >
<!ATTLIST  Cross
        Tiempo      NMTOKEN     #FIXED  "IntervalOperator"
        Id          ID          #REQUIRED
        From        IDREF       #REQUIRED
        To          IDREF       #REQUIRED
        Delay1      CDATA       "[0:0,?:0]:d"
        Delay2      CDATA       "[0:0,?:0]:d"
>

<!ELEMENT  Overlaps  EMPTY >
<!ATTLIST  Overlaps
        Tiempo      NMTOKEN     #FIXED  "IntervalOperator"
        Id          ID          #REQUIRED
        From        IDREF       #REQUIRED
        To          IDREF       #REQUIRED
        Delay1      CDATA       "[0:0,?:0]:d"
        Delay2      CDATA       "[0:0,?:0]:d"
        Delay3      CDATA       "[0:0,?:0]:d"
>


<!-- Reaktionsoperator -->

<!ELEMENT  Reaction  (SequentialAction | ParallelAction | Activate | Deactivate | Start | Stop |
                      JumpTo | JumpRelative | Pause | Continue | Freeze | Thaw |
                      AddToSpeed | MultiplySpeed | SetSpeed | Forward | Backward )>
<!ATTLIST  Reaction
        Tiempo          NMTOKEN  #FIXED  "Reaction"
        Id              ID       #REQUIRED
        IntManager      IDREF    #REQUIRED
        PreCondition    CDATA    #IMPLIED
        PostCondition   CDATA    #IMPLIED
>


<!-- Aktionen -->

<!ELEMENT  ParallelAction  (SequentialAction | ParallelAction | Activate | Deactivate | Start | Stop |
                            JumpTo | JumpRelative | Pause | Continue | Freeze | Thaw |
                            AddToSpeed | MultiplySpeed | SetSpeed | Forward | Backward )+ >
<!ATTLIST  ParallelAction
        Tiempo      NMTOKEN     #FIXED  "CompositeAction"
>
```

```
<!ELEMENT SequentialAction (SequentialAction | ParallelAction | Activate | Deactivate | Start | Stop |
                            JumpTo | JumpRelative | Pause | Continue | Freeze | Thaw |
                            AddToSpeed | MultiplySpeed | SetSpeed | Forward | Backward )+ >
<!ATTLIST SequentialAction
        Tiempo      NMTOKEN     #FIXED "CompositeAction"
>

<!ELEMENT Activate EMPTY >
<!ATTLIST Activate
        Tiempo      NMTOKEN     #FIXED "SimpleAction"
        Element     IDREF       #REQUIRED
>

<!ELEMENT Deactivate EMPTY >
<!ATTLIST Deactivate
        Tiempo      NMTOKEN     #FIXED "SimpleAction"
        Element     IDREF       #REQUIRED
>

<!ELEMENT Start EMPTY >
<!ATTLIST Start
        Tiempo      NMTOKEN     #FIXED "SimpleAction"
        Element     IDREF       #REQUIRED
>

<!ELEMENT Stop EMPTY >
<!ATTLIST Stop
        Tiempo      NMTOKEN     #FIXED "SimpleAction"
        Element     IDREF       #REQUIRED
>

<!ELEMENT JumpTo EMPTY >
<!ATTLIST JumpTo
        Tiempo      NMTOKEN     #FIXED "SimpleAction"
        Element     IDREF       #REQUIRED
        Instant     CDATA       #REQUIRED
>

<!ELEMENT JumpRelative EMPTY >
<!ATTLIST JumpRelative
        Tiempo      NMTOKEN     #FIXED "SimpleAction"
        Element     IDREF       #REQUIRED
        Delta       CDATA       #REQUIRED
>

<!ELEMENT Pause EMPTY >
<!ATTLIST Pause
        Tiempo      NMTOKEN     #FIXED "SimpleAction"
```

```
        Element     IDREF       #REQUIRED
>

<!ELEMENT Continue EMPTY >
<!ATTLIST Continue
        Tiempo      NMTOKEN     #FIXED "SimpleAction"
        Element     IDREF       #REQUIRED
>

<!ELEMENT Freeze EMPTY >
<!ATTLIST Freeze
        Tiempo      NMTOKEN     #FIXED "SimpleAction"
        Element     IDREF       #REQUIRED
>

<!ELEMENT Thaw EMPTY >
<!ATTLIST Thaw
        Tiempo      NMTOKEN     #FIXED "SimpleAction"
        Element     IDREF       #REQUIRED
>

<!ELEMENT AddToSpeed EMPTY >
<!ATTLIST AddToSpeed
        Tiempo      NMTOKEN     #FIXED "SimpleAction"
        Element     IDREF       #REQUIRED
        Delta       CDATA       #REQUIRED
>

<!ELEMENT MultiplySpeed EMPTY >
<!ATTLIST MultiplySpeed
        Tiempo      NMTOKEN     #FIXED "SimpleAction"
        Element     IDREF       #REQUIRED
        Factor      CDATA       #REQUIRED
>

<!ELEMENT SetSpeed EMPTY >
<!ATTLIST SetSpeed
        Tiempo      NMTOKEN     #FIXED "SimpleAction"
        Element     IDREF       #REQUIRED
        Speed       CDATA       #REQUIRED
>

<!ELEMENT Forward EMPTY >
<!ATTLIST Forward
        Tiempo      NMTOKEN     #FIXED "SimpleAction"
        Element     IDREF       #REQUIRED
>
```

```
<!ELEMENT  Backward  EMPTY >
<!ATTLIST  Backward
          Tiempo      NMTOKEN      #FIXED  "SimpleAction"
          Element     IDREF        #REQUIRED
>


<!-- Auswahlgruppe -->

<!ELEMENT  SelectionGroup  (Alternative+) >
<!ATTLIST  SelectionGroup
          Tiempo      NMTOKEN              #FIXED  "SelectionGroup"
          Id          ID                   #REQUIRED
          Policy      (first | static | dynamic) "static"
>

<!ELEMENT  Alternative  EMPTY >
<!ATTLIST  Alternative
          Tiempo      NMTOKEN      #FIXED  "Alternative"
          Id          ID           #REQUIRED
          Priority    CDATA        "0"
          Elements    IDREFS       #IMPLIED
>
```