Synchronization Aspects of a Presentation Model for Hypermedia Documents with Composite Nodes

Guido Lemos de Souza Filho
DIMAp, UFRN
guido@dimap.ufrn.br

Luiz Fernando G. Soares
lfgs@inf.puc-rio.br

Marco Antonio Casanova
Centro Científico Rio, IBM Brasil
casanova@vnet.ibm.com

PUC-RioInf.MCC31/95 October, 1995

Abstract: This paper outlines the synchronization aspects of a hypermedia presentation model, called Nested Context Presentation Model (NCPM), which has three major features. First, it allows authors to specify temporal synchronization constraints among events of interest within hypermedia documents. Second, it allows asynchronous material, such as user interaction links, or programs, to be combined with synchronous material in a single hypermedia document. Third, it separates the presentation specification from the structural definition of the hypermedia document. The synchronization facilities of NCPM can be applied to any hypermedia model with composite nodes and, with minor simplifications, even to those conceptual models without composites. In particular, the synchronization facilities can be applied to those models that can be translated to MHEG objects.

Keywords: spatial synchronization, temporal synchronization, presentation model, hypermedia document, NCPM

Resumo: Este relatório descreve os aspectos relativos à sincronização de um modelo de apresentação de documentos hipermídia, chamado Nested Context Presentation Model (NCPM), que possui três caracterísiticas principais. Primeiro, ele permite que os autores especifiquem restrições de sincronização temporal entre eventos relevantes que ocorrem durante a apresentação de documentos hipermídia. Segundo, ele permite que elementos assíncronos, como elos disparados via interação com usuários, ou execuções de programas, sejam combinados com elementos síncronos em um único documento hipermídia. Terceiro, ele separa a especificação da apresentação da definição estrutural do documento hipermídia. As facilidades de sincronização do NCPM podem ser aplicadas a qulaquer modelo hipermídia que possua nós de composição e, com pequenas simplificações, mesmo aos modelos conceituais que não possuem composições. Em particular, as facilidades de sincronização podem ser aplicadas aos modelos que podem ser traduzidos em objetos MHEG.

Palavras-chave: sincronização temporal, sincronização espacial, modelo de apresentação, documento hipermídia, NCPM

1. Introduction

A structural hypermedia conceptual model treats a document as an essentially passive data structure. A hypermedia system must, however, provide tools for the user to access, browse, manipulate and navigate through the network structure. This functionality is captured by the presentation model. This paper focuses on the synchronization aspects of a hypermedia presentation model that has two major features. First, it allows the authors to specify temporal synchronization relationships among events of interest within multimedia objects. Second, it allows asynchronous material, such as user interaction links, or programs, to be combined with richly coordinated synchronous material in a single hypermedia document.

The Nested Context Presentation Model (NCPM) presented in this paper addresses both the synchronization and maintenance problems. Although developed in the HyperProp project[1], which uses the Nested Context Model as its structural model, the synchronization facilities of the NCPM can be applied to any hypermedia model with composite nodes and, with minor simplifications, even to those conceptual models without composites. In particular, the synchronization facilities can be applied to those models that can be translated to MHEG objects. NCPM defines entities with a higher level semantic than the MHEG objects [MHEG93], but that can be coded in MHEG objects.

The paper is organized as follows. Section 2 presents the synchronization problem. Section 3 reviews hypermedia structural models with composite nodes, in those aspects which will be related to the synchronization. Section 4 extends the model to support document presentation. Section 5 discusses related work. Finally, section 6 contains the conclusions.

2. Synchronization

Synchronization in multimedia presentations [SoTC92] deals with the spatial positioning of the components that will be presented, also called spatial synchronization, and with the temporal positioning, or temporal synchronization.

The notion of spatial synchronization depends on the media and is based on operators that define how to combine objects to be presented on an output device at a given point in time. For example, the spatial synchronization of image, graphic or text objects defines how to position the objects for presentation in an application's window. The synchronization, in this case, involves operations like scale changes, cut, color conversions and spatial positioning inside the window. For audio objects, the spatial synchronization involves, for example, mixing the audio trails in the audio output device, with gain and tone adjustments.

The notion of temporal synchronization is based on mechanisms that define the objects' presentation schedule in output devices. The schedule can be natural, like during the simultaneous capture of audio and video, or synthetic, as in documents that have audio annotations. The presentation model described in this paper deals with synthetic synchronization.

Buchanan and Zellweger proposed in [BuZe93] a framework to classify multimedia systems, considering three points: the facilities the systems provide to support media segments (the data presented in documents), the methods used to define the media segments in a document and the formatting algorithm used. In the present paper, we are not interested in discussing

---

[1] The work reported in this paper was partially financed by a grant from the HyperProp-PROTEM project fo CNPq-Brazil.

formatters for hypermedia documents with temporal relationships, but rather focus on the first two points, captured by our presentation model. In the discussion that follows, we will use the terminology proposed by Buchanan and Zellweger, which we summarize in what follows.

The first point of the framework analyzes the way the models specify the presentation of a *media segment*. Four characteristics are examined: granularity, duration, flexibility and QoS (Quality of Service) metrics. The *granularity* of a media segment specification deals with the amount of internal structure of the media segment that the model makes accessible to the authors for specifying synchronization. The models with *coarse granularity* provide access only to the end points of the media segment's presentation. A model with *fine granularity* permits the manipulation of internal points that occur during the media segment's presentation. The *duration* characteristic verifies if it is possible to define the lengths of time required to prepare and present media segments. A *predictable duration* is one whose length can be determined a priori, as opposed to an *unpredictable duration* where the length of the media segment's presentation can only be determined when it ends. The *flexibility* deals with the possibility of specifying parameters to manipulate the duration and the form of the media segment's presentation. The models classified as *continuously adjustable* permit the specification of a range from which the duration of a media segment can be selected. For example, the duration of the presentation of an audio segment can vary in a range from 10 to 12 seconds, according to whether the silence intervals are shortened or not. Another way to specify the flexibility of a media segment's presentation is to permit the definition of distinct alternative representations that have different durations. An example of this kind of flexibility is a text segment that can be exhibited graphically in a display or as audio with the help of a synthesizer. The models that provide this kind of flexibility are classified as *discretely adjustable*. The *QoS metrics* provide an objective measure of the quality of a particular media segment's presentation. This information, combined with the attributes that specify flexibility, can be used to select the best representation for each media segment in a given environment.

The second point of the framework deals with the *temporal relationships* that describe how media segments should be temporally combined to produce a hypermedia document's presentation. Temporal relationships have four relevant attributes: granularity, temporal relation types, flexibility and QoS metrics. *Granularity* specifies where temporal relationships can be placed in: a point, a composite point, an interval or a composite interval. A *point* can be: an absolute time, a relative time during the document's presentation (5 seconds after the start of a media segment's presentation), a predictable or unpredictable event in a media segment's presentation, or an external event that can be passed through to the system (user interaction). A *composite point* is any temporal relationship that produces a specific point in time, for example, when the presentations of two media segments finish. An *interval* can be an entire media segment or a portion of a media segment. A *composite interval* is any temporal relationship that produces an interval of time, for example, while two media segments are being presented. The *temporal relation types* are defined according to the way they control temporal ordering of media segments. *Ordering relations* are binary relations that specify the order of occurrence of points or intervals in a document presentation.[2] *Duration relations* define relationships between the duration of intervals, for example, the duration of the presentation of the media segment A must be equal to the duration of B. The *complex relations* require programmatic capabilities and are classified in: group relations, iteration relations and conditional relations. *Group relations* allow the author to collect temporally unrelated intervals

[2] Allens in [Alle83] enumerate a set of ordering relations: before, starts, finishes, meets, equals, overlaps and during.

and points together so that they can be used as a single entity in another temporal relationship, for example, start the presentation of A when C or D or F finish. *Iteration relations* specify that an interval be presented repeatedly a specific number of times, or until (or while) a certain condition holds. *Conditional relations* are dependent on the document or system being in a particular state, such as presenting the audio if the workstation has audio hardware. The presentation models can support two forms of *flexibility* in the specification of the temporal relationships. In the first one, the models permit authors to specify that a temporal relationship is either *mandatory*, that is, must be satisfied in the presentation, or *optional*, whereby the temporal relationship is desirable but may be ignored. The second type of flexibility is based on the specification of a *range* of time values within which a temporal relationship is considered satisfied. For example, the author can use this feature to define that the presentation of two media segments must finish together with a tolerance of plus or minus 2 seconds. The models that permit the specification of *QoS metrics* provide a way to measure the degradation in presentation quality due to ignoring an optional temporal relationship or to varying a temporal relationship within a specified range.

The *other parameters* relevant to the classification of the presentation models are: behavior specification, spatial specification, workstation specification and network specification. *Behavior specification* allows authors to set a behavior characteristic on a point of the media segment's presentation. An example of this is the modification of the volume when an audio segment reaches 50% of its duration. *Spatial specification* allows authors to control the physical appearance of media segments, such as specifying the position and size of the window where an image segment will be presented. *Workstation specification* describes the capabilities of the workstation upon which a document may be presented (the hardware devices and software environment). Finally, the *network specifications* define the characteristics of the network that support the transmission of the media segments.

We will present the NCPM model taking into account all these aspects. As it is based on a hypermedia model with composite nodes, we summarize such a generic model in the next section.

3. Hypermedia Documents with Composite Nodes

This section is dedicated to present the characteristics of hypermedia models required to use the full capabilities of NCPM. We will illustrate these requirements describing, through a library of classes, a generic model that summarizes the functionalities of several models like Hyperbase [ScSt90], HyperProp [SoCR94, SoCR95], or any model based on MHEG proposed standard. The following is thus a brief description of these classes and their functionality. Moreover, we concentrate only on the entities that are relevant to the description of the presentation model.

The class hierarchy for a basic conceptual model for hypermedia documents that supports composition is illustrated in Figure 1.

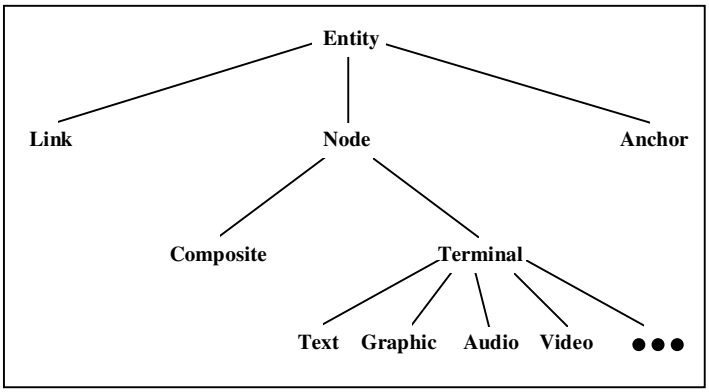

Figure 1 - Class Hierarchy For a Basic Conceptual Model

An *entity* is an object that has as attributes at least a *unique identifier* and an *access control list.* The unique identifier has the usual meaning. For each entity attribute, the access control list has an entry that associates a user or user group to his access rights for the attribute.

A *node* is an entity that has as additional attributes an *anchor list* and a *content.* The former concept is addressed in detail below and the exact definition of node content depends on the class of the node.

A *terminal node* is a node whose content and anchor list are both application dependent. The class of terminal nodes can be specialized into other classes (*text, audio, image*, etc.).

A *composite node C* is a node whose content is a set *L* of nodes and links. We say that an entity *E* in *L* is a *component of C* and that *E is contained in C.* We also say that an entity *A* is *recursively contained in B* iff *A* is contained in *B* or *A* is contained in an entity recursively contained in *B*. The class of composite nodes may be specialized into other classes not relevant in our discussion.

In a node, each element of the anchor list is called an *anchor* of the node and is an entity which has as additional attribute a *region*. Note that an anchor may belong to more than one anchor list.

We define a region of a terminal node as either the special symbol λ, representing the entire node content, or as a set of *marked information units*.[3] The exact notion of information unit and marked information unit is part of the definition of the terminal node. For example, a information unit of a video node could be a frame, while a information unit of a text node could be a character. Any subset of the information units of a terminal node may be marked.

The anchor list acts as the external interface of a terminal node *N,* in the sense that an entity can access regions of the content of *N* only through the anchor list of *N*. Hence, anchors shield other entities from changes to the content of *N*. As an example, consider a text node with just one anchor whose region points to the second paragraph of the text. Assume that a link indirectly refers to the second paragraph of the text by identifying the appropriate anchor. Then, any changes to the text must be reflected only in the anchor region and do not affect the link.

---

[3] The concept of marked information units is similar to the marked states concept introduced by Buchanan [BuZe92].

For a composite node *C*, an anchor region must be either the special symbol λ or a subset of *L* containing only nodes (λ again represents the entire node content). The anchor list of *C* has about the same role as that of a terminal node, that is, to act as an external interface to the node, except that it does not entirely shield entities from changes to the content of *C*, as explained in [CaSS95].

A *link* is an entity with two additional attributes, the *source end point set* and the *destination* or *target end point set*. The values of these attributes are sets whose elements, called *end points* of the link, are pairs of the form $<(N_k,...,N_1),A>$ such that N_1 is a node, N_{i+1} is a composite node and N_i is contained in N_{i+1}, for all $i \in [1,k)$, with $k>0$, and A is an anchor of N_1. The node N_k is called a *base node* of the link.

It is also required that, for every link *l* contained in a composite node *C,* for every end point $<(N_k,...,N_1),A>$ of *l*, node N_k must be either *C* or a node contained in *C*.

4. The Presentation Model

The Nested Context Presentation Model (NCPM), has its name derived from the environment where it was developed: the HyperProp system, which uses NCM (Nested Context Model) [SoCR94, SoCR95] as its structural model for hypermedia documents.

In NCPM, the presentation of a hypermedia document is composed of the presentations of its basic information units. An information unit in a composite node is a node contained in it. An information unit in a terminal node is a region. The exhibition of an information unit defines an *exhibition event* and the selection of an information unit by a user defines an *interactive event*. Therefore, in NCPM, a hypermedia document's presentation is decomposed as a collection of events.

Among the events that occur during a hypermedia document's presentation, only those that are related in time with others or are associated with changes in the presentation behavior are relevant in the temporal synchronization context. The relationship between two or more events, or the association of an event with a behavior change in the document's presentation, defines a *synchronization point*. The alignment of events in the synchronization points defines the temporal synchronization in the hypermedia document's presentation.

In NCPM an author can directly mark events of interest in a media item by previewing the underlying media object using interactive tools rather than by specifying timings. Furthermore, authors can modify documents by directly manipulating the events and the temporal relationships among those events rather than modifying timings.

As in the previous section, we will discuss the presentation model through a description of its classes and their functionality, as shown in Figure 2.

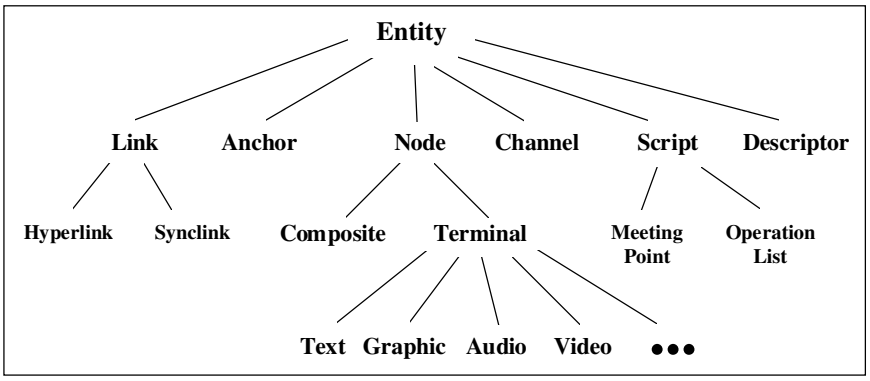

Figure 2 - Class Hierarchy of NCPM

4.1. Descriptor Objects

In the Nested Context Presentation Model, the user's interaction with a hyperdocument is modeled in conformance to the Dexter Model (DM). The fundamental concept in the runtime layer of DM is the instantiation: a presentation of the component to the user. The instantiation of a node also results in the instantiation of its anchors. An instantiation in NCPM is called a *representation object,* discussed in detail in Section 4.2. Representation objects are created from a node and its associated descriptor object.

A *descriptor* consists of a collection of event descriptions and a start-up specification. Each event description contains a region description, an operation list and a specification of the event type.

Each event description may define a synchronization point at which the display of the associated terminal or composite node can be synchronized with events in the same terminal or composite node or in others. An event description may also define a synchronization point at which the behavior of the representation of a terminal or composite node can be altered.

The regions associated to events are those that define synchronization points, and correspond either to instantiated anchors (end points of links), or to marked information units at which the behavior of the representation of a terminal or composite node can be altered.

Events are divided into two types: synchronous and asynchronous. The *synchronous event*s are those whose temporal placement relative to other events is known in advance. The *asynchronous events* are those whose time of occurrence can only be determined at runtime, for example, a user interaction.

Each descriptor has one distinguished synchronous event, marking the beginning of the display of the associated terminal or composite object. The end of the display of the associated object can be a synchronous event if we know in advance the duration of the presentation of this object, or an asynchronous event, otherwise.

Operations lists in events are used to control the display behavior of a document. An *operation list* consists of an ordered list of operations. An operation is composed by a set of actions and a set of conditions (that must be satisfied to trigger the actions).[4] The conditions cannot use in their definition the occurrence of other events, as will be seen later. Only the links can do so.

---

[4] Note that we use the words *actions* and *conditions* with the same meaning used by MHEG.

A start-up specification contains information to initiate the presentation of an entity. In particular, it defines the channels that will be used for exhibition or edition of entities. These channels, as we will see in the next section, are used to define the interface with a program, and in particular, an editor. The Nested Context model provides specific programs (and the corresponding channels) for editing system attributes, and browsing and editing the contents of composite nodes.

A start-up specification also has an ordered list of operations, which must be executed before a node begins to be shown, to prepare its exhibition. A start-up specification must define all parameters required to create a representation object.

4.2. Representation Objects

A representation object is created by the runtime display controller (the equivalent of the Dexter instantiator) from the NCPM objects. It is created from a *descriptor*, a *node*, *methods* and a *set of channels*. We say that the created object is a representation of the node. Representation objects are responsible for detecting and signaling the occurrence of events.

The descriptor can be defined in an attribute of the node from which the representation object (a version of the node) is created. Links may also have a set of descriptors, as in the Dexter model, which contains information for the presentation model indicating how a representation object of referenced nodes should be created. Composite nodes also have, for each of its component nodes, an attribute where a descriptor, for creation of the representation object of the component may be defined.

When presenting a node, the descriptor defined explicitly by the user bypasses the descriptor defined by the composite node that contains the node; this in turn bypasses the descriptor defined by the link used to reach the node, which bypasses the descriptor defined in a node attribute. Each node type has a default descriptor, which is used if none of the descriptors presented above is defined.

The *methods* operate on the terminal or composite object and its events. *Behavior specification methods* provide the functions to implement a user interface for creating and editing events in representation objects. *Time analysis methods* facilitate the automatic creation of a document schedule by providing the system with timing information about a media item, such as a duration between two events. *Control methods* allow the system to control the behavior of a media object while the document is being displayed, such as speeding up or slowing down the audio playback rate.

Each media type provides control procedures that can be used to affect the display behavior of the representation object of that type. All media types must provide procedures to start, end, pause, and resume the display of the representation object. However, a media type can also supply type-specific control procedures.

The *set of channels* is used to define the requirements in terms of multimedia resources (hardware devices or software environments) that the representation object will need to present the node content, and to provide a homogeneous interface to these resources.

4.3. Channel Objects

The channel object is an abstraction used to provide transparent access to the multimedia resources (hardware devices or software tools) used to exhibit the nodes. The channel object provides a homogeneous interface to the hypermedia system and knows how to use the

resources to play the media. The channel maps the invocations of methods produced by the representation object in the corresponding operations in the hardware devices, directly, using operating systems calls, or indirectly, using an intermediate software, like an editor, for example. The channel is also responsible for the inverse communication, that is, to pass information generated by the multimedia resources to the representation objects, as the regions presented to or selected by the user.

4.4. Reinterpreting Links

At runtime, a *link* is now reinterpreted as a logical assertion about a collection of events. Links define relationships among nodes, such as *go to* relationships (for reference) and synchronization relationships (such as those defined in the MHEG proposal). Multiple source and destination end points allow the definition of many-to-many relations, which is intended to support applications where, for example, the selection of a link can lead to the simultaneous exhibition of several nodes, if a series of conditions in the source end points is satisfied. Each of the *end points of a link* is now redefined by a (possibly unary) list of representation nodes (N_k,..., N_2, N_1) and one event, which must belong to the set of events of N_1. The node N_{i+1} must be a representation of a composite node and N_i must be contained in N_{i+1}, for all $i \in [1,k)$. The node N_k is called a *base* of the link. The list of nodes in each end point allows the definition of links connecting information not directly contained in the same composite.

Many hypermedia systems implement links by means of tags and pointers embedded in documents. In NCPM there are two worlds of information: the world of raw information: resource bases of text, graphics, video and sound; and the world of relationships and links between ideas, words, phrases, images and documents. NCPM aims at keeping these separate. Links are contained in composite nodes and are not embedded in media objects.

A *hypermedia link* (from here on called hyperlink) is expressed as a relation between a set of events in one or more nodes, represented by the source end points, and one or more events in the same or other nodes, represented by the target end points. At least one of the events of the source end points must be an asynchronous event defined by a user interaction.

A *synchronization link* (from here on called synclink) is expressed as a relation between a set of events in one or more nodes (synchronous events or asynchronous events that are not defined by a user interaction), represented by the source end points, and one or more events in the same or other nodes, represented by the target end points.

A link may also contain two special attributes: one, already mentioned, associates a descriptor to each of its target end points, and another called meeting point.

Relations associated with a link (synclink and hyperlink) are expressed through its *meeting point*, which consists of an ordered list of operations, composed of a set of conditions and a set of actions. The conditions are based on the source end points. Whenever a condition is satisfied it triggers the associated actions. Actions of a meeting point are operation lists and may trigger operation lists defined in the descriptor objects that contains the target end points.

4.5. Links, Events and Messages

The semantics of links and events defined in NCPM can be implemented with message passing.

As the events of a node object occur by the exhibition or selection of their associated anchors, they are processed by the representation object that instantiates the node. To process an event is to verify, in its operation list, which of the operations are enabled (the associated condition is

satisfied) and to run these operations. Two examples of conditions are: (event.state = on) and (event.state = off). To execute an operation is to dispatch a message invoking the method activation in the target object. The target of the message can be the source object itself or a link.

NCPM links can be implemented dispatching messages "*occurred*" from the source end points to the link, and dispatching messages "*process*" from the links to the target end points. The operation list of the source points of a link *L* must contain an operation that, when processed, generates a message "*occurred*" to the representation object that control the presentation of the composition node *C*, where the link is defined. The representation object *C*, then, passes the message to the link *L*.

When the link *L* receives messages "*occurred*" from its source end points, it checks the rules defined in its meeting point. When the result of the rule evaluation is successful, *L* sends messages to the representation objects R_i that contains the target end points of the link *L*, associated with the satisfied rule. Messages "*set*" and "*reset*" are used to attribute the values "*on*" and "*off*" to the states of the target events of *L* — the target events are not processed when their state is modified. The link *L* must send a message "*active"* to R_i, when it wants the operation list of the target end point to be processed. Other messages can be sent (for example, the message "*show*"*)* that cause the exhibition of the content of the node R_i, beginning in the region associated to the target end point. In any case, if R_i has not been instantiated, *C* must call for its instantiation. It is important to remember that a target end point can be a list of representation nodes (N_k,..., N_2, N_1) and one event. In this case, all the composite nodes containing the base node of the event that have not yet been instantiated must be instantiated when the link is traversed.

Figure 3 exemplifies the functioning of synclinks. This figure shows three representation objects (RO_1, RO_2 and RO_C), the source end point of the synclink (E_2 in RO_1), the target event (E_5 in RO_2) and the meeting point (*PE*). In this case, *PE* defines that the target end point must be processed when the event associated with the source end point occurs.

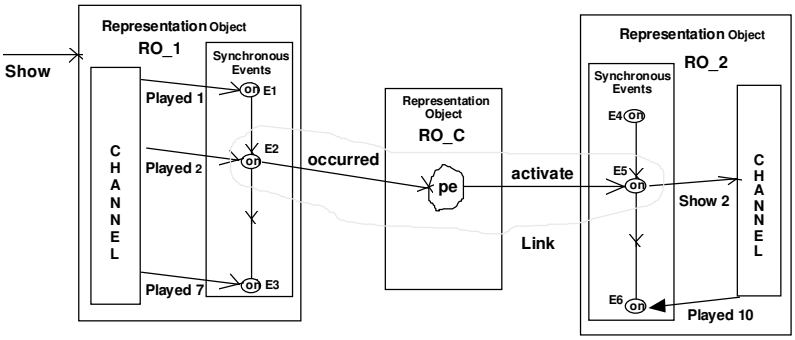

Figure 3: Synclink

When the source event E_2 is processed by the representation object RO_1, its operation list is ran and the operation associated with the condition (event.state = on) is executed. The execution causes the dispatching of a message "*occurred*" to RO_C (the representation object that contains the link). In the meeting point attribute of the link, a rule defines that a message to the target of the link must be sent immediately. This makes RO_C to send a message "*activate*" to RO_2. When RO_2 receives this message it processes E_5. In the operation list of E_5, the operation "*show region 2*" is, for example, conditioned to the *on* state of the event. As this

is the current state of E_5, the operation is executed, and commands the channel associated to RO_2 to begin the exhibition of the node content from the region 2.

Figure 4 shows how synclinks can be used for temporal alignment of events to maintain the temporal synchronism within an acceptable quality. In the alignment shown in this figure, each source synchronization point is followed by a target synchronization point. When the source events of the link are processed (E_2 or E_6), messages "*occurred*" are sent to RO_C, and messages "*pause*" are sent to the respective channels. The rule in the meeting point attribute of the link has, as condition, the logical expression ($E_2 \wedge E_6$). When this expression is satisfied, RO_C sends messages "*activate*" to the target end points, which process their operation lists. The result of this processing is the dispatching of messages "*resume*" to the associated channels.

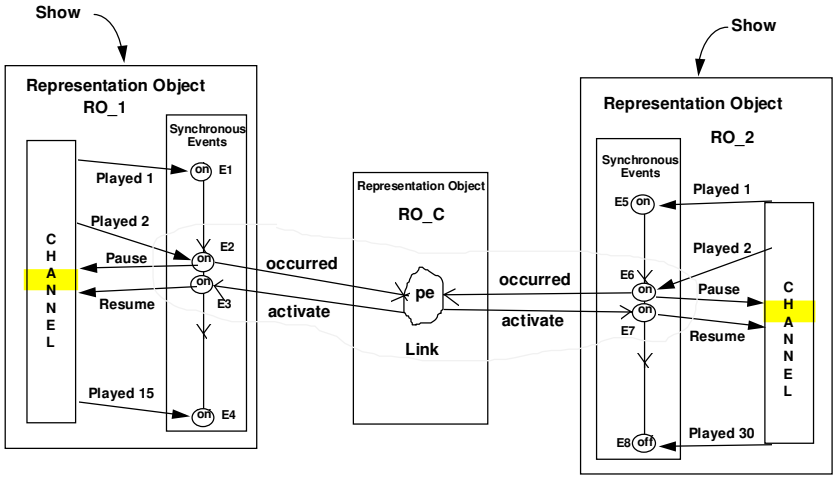

Figure 4 - Temporal alignment of events

Imagine that in the preceding case the event E_2 occurred before E6. In this situation, another rule of the meeting point, defined by the expression ($E_2 \wedge \neg E_6$), can be used to program the dispatching of a message to RO_2 commanding the speed-up of the exhibition of the content of the node. The same rule should enable the dispatching of another message to RO_1, requiring that it reduce the speed of exhibition, as illustrated in Figure 5.

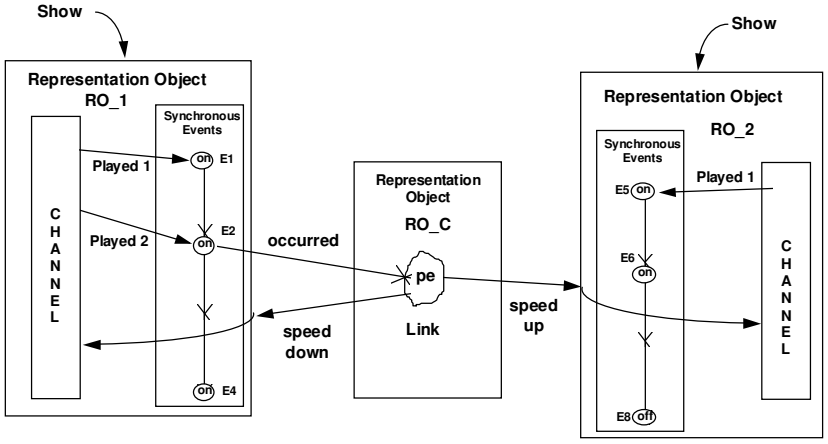

Figure 5 - Speed adjusts in the exhibition of a node

Hyperlinks differ from synclinks as at least one of the events associated to the source end points must be an asynchronous event, defined by a user interaction, as illustrated in Figure 6. In this figure, the object RO_1 commands the beginning of the exhibition of its associated node. Anchor 2 is associated with the events E_2 and E_4. The event E_2 is an exhibition event and E_4 is an interactive event. This double association can be used, for example, to program the trigger of a beep, in the operation list of E_2 to call the user attention to the anchor. The user selection of this anchor implies in the processing of E_4. As a consequence, the message "*occurred*" is sent to RO_C. From this point on, the functionality is identical to the synclink previously ilustrated.

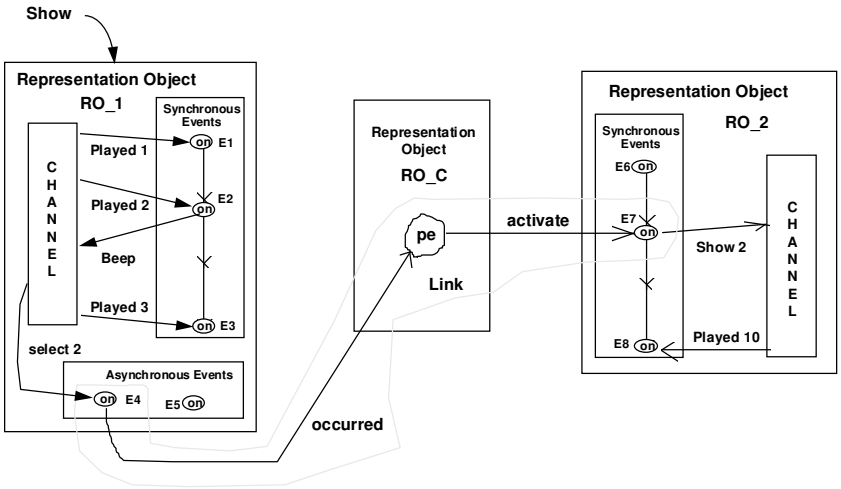

Figure 6 - Hyperlink

4.6. Scripts

The NCPM model allows the temporal and spatial synchronism to be optionally specified by scripts (treated by the model as an operation list). The model defines the script object, whose function is to encapsulate a program written in a language whose instructions (operations), when executed, generate messages invoking the activation of valid methods in the NCPM representation objects.

The operation list and meeting point classes are specialization of the class script. As part of the HyperProp project, a language was developed to test the functionality of the script object, denominated SDD (Show Dynamic Description) [SoSo95].

5. Related Work

Table 1 shows some examples of presentation models classified within the framework proposed by Buchanan and Zellweger, resumed in the Section 2. This table updates a previous one presented in the same reference including the NCPM model. The examples presented are: Temporal glue [HaSR92], Trellis [StFu90], Geneva temporal scripting language [FiTD87], CMIFed [BuRL91, RJMB93], MODE [BlHL92], Firefly [BuZe93] and NCPM.

Tabela 1: Presentation models comparison using the framework defined by Buchanan and Zellweger

		Temporal Glue	Trellis	Geneva	CMIFed	MODE	Firefly	NCPM
Media Segments	Granularity							
	Coarse	+	+	+	+	+	+	+
	Fine	+	-	+	-	+	+	+
	Duration							
	Predictable	+	+	+	+	+	+	+
	Unpredictable	-	-	-	-	+/-	+	+
	Flexibility							
	Continuous	+/-	-	-	-	+	+	+?
	Discrete	-	-	-	-	+	-	+?
	QoS Metrics	?	-	-	-	+	+	+?
Temporal Specification	Granularity							
	Points	+	-	+	+	+	+	+
	Intervals	-	+	+	+	-	-	+
	Composites	-	-	+	+	-	-	+
	Relations							
	Ordering	+	+	+	+	+	+	+
	Duration	-	-	-	-	-	-	-
	Complex	-	+	+	-	-	-	+/-
	Flexibility							
	Mandatory/Optional	-	-	-	+	-	-	+?
	Range	-	-	-	+	-	-	+?
	QoS Metrics	-	-	-	-	-	-	+?
Other Parameters	Behavior specification							+
	Spatial specification							+
	Workstation specification							+/-
	Network specification							-

+ available
+/- more or less
- not available
+? available but not explained in this paper

6. Conclusion

This paper presented the Nested Context Presentation Model throught the definition of a set of basic objects that can be used to model all the relevant aspects regarding temporal and spatial synchronization in the presentation of a hypermedia document.

The idea behind NCPM design was to provide high level abstractions to define synchronism in the document's presentation, considering the possible use of the syntactic structures provided by the MHEG objects.

A full implementation of an environment that uses the NCPM as presentation model is under development as part of the HyperProp project.

References

[Alle83] Allen, J. "Maintainnig knowledge about temporal intervals". *Commun. ACM*. November 1983.

[BlHL92] Blakowiski, G.; Hubel, J. and Langrehr, U. "Tool Support for the Synchronization and Presentation of Distributed Multimedia". *Computer Communications*. December 1992.

[BuRL91] Bulterman, D.; van Rossum, G. and van Liere, R. “A Structure for Transportable, Dynamic Multimedia Documents”. *In Proceedings 1991 Summer USENIX Conference*. June 1991.

[BuZe92] Buchanan, M.C.; Zellweger, P.T. "Specifying Temporal Behavior in Hypermedia Documents", *Proceedings of European Conference on Hypertext, ECHT'92.* Milano. December 1992.

[BuZe93] Buchanan, M.C.; Zellweger, P.T. “Automatic Temporal Layout Mechanisms” *ACM Multimedia 93.* California. 1993.

[CaSS95] Casanova, M.A.; Souza, G.L.; Soares, L.F.G. "Anchors and Links for Composite Nodes ". *Research Report Departamento de Informática, PUC-Rio*. Rio de Janeiro, Brasil. Setembro de 1995.

[FiTD87] Fiume, E. Tsichritzis, D. and Dami, L. “A Temporal Scripting Language for Object-oriented Animation”. *In Proceedings Eurographics’87*. Elsevier Science Publishers. North-Holland Publishing Company, Amsterdan, 1987.

[HaSc90] Halasz, F.G.; Schwartz, M. "The Dexter Hypertext Reference Model". *NIST Hypertext Standardization Workshop*. Gaithersburg. January 1990.

[HaSR92] Hamakawa, R; Sakagami, H. and Rekimoto, J. “Audio and Video Extensions to Graphical User Interfaces Toolkits”. *In Proceedings Third International Workshop on Network and Operating System Support for Digital Audio and Video*. San Diego, CA. November 1992.

[MHEG93] MHEG. "Information Technology - Coded Representation of Multimedia and Hypermedia Information Objects - Part 1: Base Notation. *Committee Draft ISO/IEC CD 13522-1*. July 1993.

[RJMB93] van Rossum, G.; Jansen, J.; Mullender, K.S. and Bulterman D. “CMIFed: A Presentation Environment for Protable hypermedia Documents”. *Proceedings ACM Multimedia’93*. Anaheim, CA. August 1993.

[ScSt90] Schütt, H.A.; Streitz, N.A. "HyperBase: A Hypermedia Engine Based on a Relational Database Management System". *Procceedings of European Conference on Hypertex, ECHT'90. 1990.*

[SoCC93] Soares, L.F.G.; Casanova, M.A.; Colcher, S. "An Architecture for Hypermedia Systems Using MHEG Standard Objects Interchange". *Information Services & Use*, vol.13, no.2. IOS Press. Amsterdam, The Netherlands. 1993; pp. 131-139.

[SoCR94] Soares, L.F.G.; Casanova, M.A.; Rodriguez, N.L.R. "Nested Composite Nodes and Version Control in Hypermedia Systems". *Proceedings of the Workshop on Versioning in Hypertext Systems*, in connection with ACM European Conference on Hypermedia Technology, Edinburgh. Setembro de 1994.

[SoCR95] Soares, L.F.G.; Casanova, M.A.; Rodriguez, N.L.R. "Nested Composite Nodes and Version Control in an Open Hypermedia System". *IEEE Transaction on Information Systems; Special issue: Multimedia Information Systems. (Accepted for publication).*

[SoSo95] Souza, G.L.; Soares, L.F.G. "Synchronization Aspects of the Nested Context Hypermedia Presentation Model". *Research Report Departamento de Informática, PUC-Rio*. Rio de Janeiro, Brasil. Setembro de 1995.

[SoTC92] Soares, L.F.G.;Tucherman, L.; Casanova, M.A.; Nunes, P.R. "Fundamentos de Sistemas Multimídia". *Porto Alegre: Instituto de Informática da UFRGS.* 1992.

[StFu90] Stotts, D. and Furuta, R. “Temporal Hyperprogramming”. *J. Visual Languages and Computing*. September 1990.