

HYPERMEDIA SPATIO-TEMPORAL SYNCHRONIZATION RELATIONS ALSO DESERVE FIRST-CLASS STATUS

DÉBORA CHRISTINA MUCHALUAT-SAADE, LUIZ FERNANDO GOMES SOARES

TeleMídia Laboratory – Departamento de Informática – PUC-Rio

R. Marquês de São Vicente, 225 – Gávea – 22453-900, Rio de Janeiro, RJ, Brazil

debora@telemidia.puc-rio.br, lfgs@inf.puc-rio.br

This paper introduces the concept of hypermedia connectors, presenting how spatio-temporal synchronization relations may be defined as a specialization of architecture description language (ADL) connectors. Some features found in ADLs were brought to hypermedia authoring languages, giving first-class status to relations expressed by links. Among the profits, we can highlight the possibility of reusing a spatio-temporal relation specification in different links with the same behavior and the possibility of defining composite spatio-temporal relations expressed by links, which is an original feature in hypermedia authoring languages.

1 Introduction

Architecture Description Languages – **ADL** – are formal languages that can be used for representing a software system architecture, defining its components and their behavior specifications, as well as patterns and mechanisms for interactions among them [14]. Usually, the building blocks of an architectural description are [8]:

- components: computation or data storage units that can be as small as a unique procedure or as big as a whole application;
- connectors: used for modeling interactions among components and rules that govern these interactions;
- architectural configurations: connected graphs of components and connectors that describe the architecture structure. Hierarchical composition is desirable in ADLs, as it allows system description in different levels of detail. A complex structure with a complex behavior may be explicitly represented or may be abstracted by a single component or connector.

In several ADLs, connectors play an important role in software architecture description. They are treated as first-class entities [13, 7] and their power of abstraction is compared to those of components.

There are some advantages of treating connectors as first-class entities, such as:

- Independence – Definition of connectors independent of the components that might interact using them.
- Reuse – Since they have an independent definition, different configurations may use the same connector. Usually a configuration defines instances of

components and connectors and specifies which components are attached to which connectors.

- Composition – Composite connectors represent groups of several connectors and components, modeling more complex interactions among components of a software architecture.

Software architecture structures are similar to hypermedia document structures. At first glance, it may look trivial to make a direct correspondence between the basic structural elements found in ADLs and the ones found in **H**ypermedia **A**uthoring **L**anguages – called **HAL** from here on. Components are analogous to nodes, connectors to links and configurations to composite nodes.

However, observing more closely, we should note that the analogy is not so direct and trivial. There are some interesting structural characteristics of ADLs that have no analogy in HALs and vice-versa. Reference [9] discusses the main differences between them, presenting a detailed comparison among some ADLs and HALs found in the literature.

The main differences between ADLs and HALs are usually related to connectors. They can be compared to hypermedia relations represented by links. These relations do not have the same treatment nodes have in HALs, because of the following limitations:

- a relation definition is usually dependent on the nodes that are related;
- a hypermedia node can be reused in different compositions, but this is not the case of relations expressed by links in the majority of HALs. In some of them, links are embedded in the node content preventing node reuse without bringing the link together, such as in HTML. In some others, links are contained in link sets, which can be stored in composite nodes, as in NCL [2]. In this case, only composite nodes can be reused, the link set alone cannot. Link sets can also be stored in independent repositories, as in the link bases of Microcosm [3] and XLink [19], for example. In this case, different link sets can be associated to the same set of documents, but the specification of a relation expressed by a link cannot be reused;
- some HALs offer composite nodes, but no one offers links representing relations whose semantics are given by compositions of other nodes and links.

The goal of this work is to give to hypermedia spatio-temporal relations, expressed by links, the same treatment software connectors have in ADLs, bringing the features provided by ADL connectors to HALs. Hypermedia spatio-temporal relations may also be represented by special composite nodes, such as SMIL [15] and NCL parallel and sequential compositions, however, this paper only focuses on expressing relations using links. Nevertheless, this is not a limitation, as relations represented by composite nodes can be translated to relations represented by links, as discussed in [12]. For a discussion about the use of compositions or links to represent synchronization relations, one may refer to [17].

Besides the mentioned comparison between ADLs and HALs, [9] also defined a structural meta-model that can be used to represent both software architecture and hypermedia structures. Although the meta-model already provides the concept of connector, it is only concerned about structural definitions. For a complete description of hypermedia spatio-temporal relations, a semantic model is also needed. The present paper uses events as the basis for specifying semantics of synchronization relations expressed by links, making this proposal generic for HALs.

The paper is organized as follows. Section 2 presents a brief description of the structural meta-model proposed in [9]. Section 3 shows how the meta-model can be specialized to represent hypermedia document structures. Section 4 presents a detailed definition of spatio-temporal hypermedia connectors. Section 5 discusses the approach presented and compares this proposal to some related work. Finally, Section 6 is dedicated to final remarks.

2 The Structural Meta-Model

The structural meta-model that can be used to represent software architecture and hypermedia document structures is a special case of compound graph.

The meta-model defines two types of vertices, called *component* and *connector*, each one having a set of *interface points*.

The meta-model also defines a basic type of arc, called *bind*, connecting a component interface point to a connector interface point or vice-versa. Binds must not directly connect component to component or connector to connector [9].

A simple structure defined by the meta-model is illustrated in Figure 1.

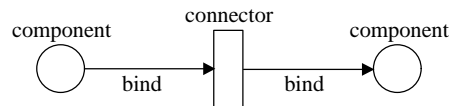


Figure 1. Example of a simple structure

Like in a compound graph, vertices may represent subgraphs also composed by components, connectors and binds, where vertices may be nested to any depth, as illustrated in Figure 2.

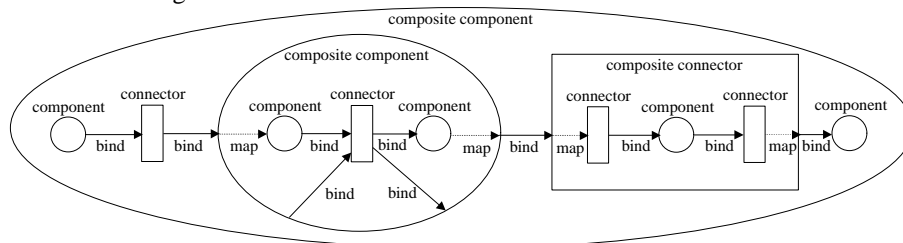


Figure 2. Composite vertices, binds and maps

Binds can only be defined between vertices having the same parent or between parent and child vertices. In order to allow indirect connections between vertices inside and vertices outside a composition, another type of arc is defined, called *map*. A map connects parent and child vertices of the same type, such as a parent component/connector interface point to a child component/connector interface point. Indirect connections can be done by an arc path containing maps. The use of binds and maps is illustrated in Figure 2. It is important to note that maps do not represent interactions between components or between connectors. They are just means for exporting internal interface points to the outside of a composite vertex.

When the structural meta-model is applied to a particular domain, some constraints will probably have to be defined in order to ensure consistency of the structure created. For example, in a hypermedia document represented by a composite component, as we shall see in the next section, for every constituent connector, there must be at least two binds attaching this connector to components.

3 Representing Hypermedia Document Structures using the Meta-Model

In order to apply the structural meta-model to the hypermedia domain, hypermedia model entities [5], such as media nodes, composite nodes and links, must be described in terms of the meta-model entities.

Media and composite nodes found in HALs are special types of components. Media node interface points are node anchors. Composite node interface points can be node anchors or node ports, as follows.

As usual, an anchor represents a marked region in the node content. In the case of a media node, an anchor can be any set of information units of its content. An information unit depends on the media node type and can be a character for a text media node, a frame for a video media node, etc. In the case of a composite node, an anchor may be any subset of its constituents. It is important to maintain anchors for hypermedia composite nodes, since a link might touch a composite node itself, for example, to start the presentation of its structure and not the presentation of its constituents.

A composite node port is used to create entry and exit points of a composite node allowing external links to touch nodes contained inside a composite node without losing compositionality. This allows, among other things, the presentation of a composition to be seen and understood as the presentation of its constituents. Through the use of maps, a composite node can make an interface point of a constituent node visible for any relation with any node outside the composite node, associating that constituent node interface point to a composite node port. An example of the use of maps inside hypermedia composite nodes will be given in Section 4.

Although several hypermedia models treat links as first class entities, such as the Dexter model [5], the AHM model [4], the NCM model [16] and the XML

Linking Language – XLink [19], relations expressed by links do not receive the same treatment. Thus, a new entity must be introduced for capturing the concept of connector. This entity was called *hypermedia connector*.

A hypermedia connector represents a spatio-temporal synchronization relationship that will be used to create hypermedia links among nodes. It specifies the relation semantics, but does not specify node anchors that will participate in the relationship. Node anchors will be specified by hypermedia links using that relation, that is, using the connector.

A connector is defined by a set of interface points, called *roles*, and a *glue* [1]. Each connector role specifies the duty of a participant of the interaction, and the connector glue describes how participants interact. Section 4 will give a complete definition for role and glue semantics based on the concept of event.

A link references a hypermedia connector, and also defines a set of binds relating the connector roles to node interface points, which may be anchors or ports. If the node interface point specified in a bind is a composite node port, there must be a map path relating that port to constituent node interface points, until an anchor is reached. The complete definition of a link is obtained by:

- a reference to a hypermedia connector;
- a set of binds relating the connector roles to node interface points;
- a set of map paths, in the case of binds to composite node ports, relating composite node ports to internal node interface points, defining sequences of nested composite nodes until an anchor of the most internal node in each sequence is reached.

A simple example of a hypermedia spatio-temporal relation is the traditional hypermedia link, exemplified by HTML links, which causes the navigation to a target node anchor when a source node anchor is selected by a user. In this case, a hypermedia connector would define two roles, identified by *selection_condition* and *presentation_action*, for example. The connector glue would give the causal semantics between the roles, specifying that if a selection happen in the participant playing the *selection_condition* role, the presentation of the participant playing the *presentation_action* role must be fired. Figure 3 illustrates a hypermedia connector *CON1* modeling the traditional hypermedia relation and document *COMPI* having two different links referencing the same connector. Link l_1 specifies that if anchor m of node A is selected, anchor I of node B is presented, and link l_2 specifies that if anchor n of node A is selected, anchor I of node C is presented.

In this section, we presented some basic concepts that can be applied to any HAL willing to give to relations first-class status. The previous example illustrated one simple semantics for a connector (the HTML link semantics). However, in order to present the definition of hypermedia connector in details, a semantic model is required, as will be discussed in the following section.

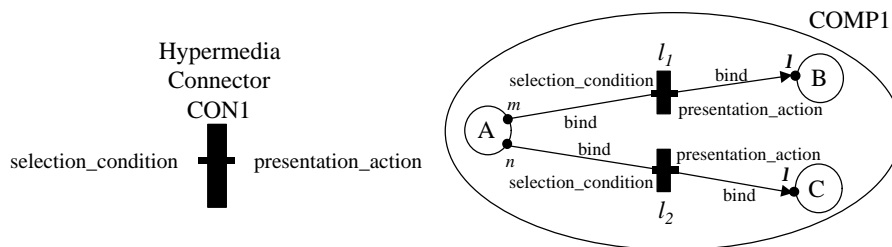


Figure 3. Example of different links using the same hypermedia connector

4 Spatio-Temporal Hypermedia Connectors

We are going to use the concept of event for specifying synchronization relationships among nodes expressed by hypermedia links. The concept of connector was used because link semantics in the majority of HALs can be expressed by transitions in an event state machine.

Event is the basic synchronization unit for specifying relationships among nodes in a hypermedia document. As stated in [10], an *event* is an occurrence in time that can be instantaneous or can occur over some time period. Some HALs specifies two types of events: *presentation event*, which is defined by the presentation of an anchor of a node and *selection event*, which is defined by the selection of an anchor of a node. Other HALs, like NCL, specify other types of events, as the *attribution event*, which is defined by the change of a node attribute value.

Event states and transitions are used for defining synchronization relationships among nodes. For example, a presentation event might be defined by the event state machine shown in Figure 4.

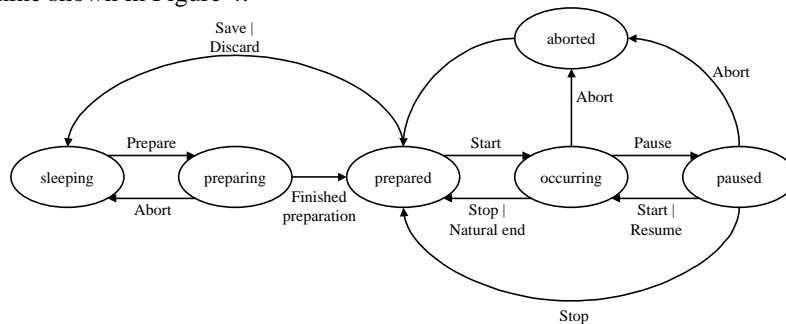


Figure 4. Example of a presentation event state machine

Not all hypermedia models have an event state machine with the granularity specified in Figure 4. In some models, the presentation event state machine has only

the *prepared*, *occurring* and *paused* states, and in others, events are confined to states *prepared* and *occurring*. An event may have an associated attribute, called *occurrences*, which counts how many times the event occurs, that is, how many times its state transits from *occurring* to *prepared* in the example shown in Figure 4. Presentation events may also have an attribute named *repeat*, which indicates how many times the event must sequentially occur every time it is started. This attribute may be assigned a finite value or the indefinite (∞) value.

Hypermedia connectors specify relationships among events, having the following attributes:

- *set of roles*, where each role defines how an event type participates in the relation, specifying conditions and actions in a causal relation, for example;
- *glue*, which defines how connector roles interact.

Hypermedia links can be used to express causal or constraint relationships among events. In causal relationships, some action is executed when a specific condition is satisfied. An example is “if node *A* is being presented (condition) and node *B* has finished its presentation (condition), present node *C* (action).” Other relationships represent constraints among events. For example, a constraint that specifies that two nodes must finish their presentation together. Note that there is no causality involved. The occurrence of the presentation of one node without the occurrence of the presentation of the other also satisfies the constraint, that only specifies that, if and only if these two nodes are presented, their end times have to coincide.

In order to provide the necessary semantics for defining causal or constraint relationships among events, a hypermedia connector may be specialized in a causal or constraint connector. A causal hypermedia connector has two types of roles: condition and action. A constraint hypermedia connector has just one type of role, condition.

A *condition role* has the following attributes:

- *id*, which must be unique among the connector set of roles;
- *event type*, which may be presentation, selection, attribution, etc.;
- *cardinality* [*min*, *max*], where $min \geq 1$, $max \geq 1$ and $max \geq min$ (*max* may be ∞). This attribute defines the minimal and maximal number of nodes that may play this condition role when the connector is used. Its default value is [1,1];
- *logical expression* of binary simple conditions or compound conditions, as follows.

Every *binary simple condition* is expressed by two unary conditions: a *previous condition*, to be satisfied immediately before the condition evaluation instant, and a *current condition*, to be satisfied at the condition evaluation instant. A binary simple condition is satisfied when both unary conditions are satisfied. A unary condition is a comparison concerning the event state value, the occurrences or the repeat attribute value of the event, using operators =, \neq , <, \leq , > or \geq . It can also receive the *true* value, if it is not relevant in the binary simple condition evaluation. A

compound condition consists of a logical expression involving other conditions, simple or compound, testing attribute values of the same event and based on the operators \wedge (and), \vee (or), \neg (not). For example, in order to specify that an event happened for the third time, the logical expression would be $((state=occurring, state=prepared) \dot{U} (true, occurrences=3))$. The main reason for specifying the previous and current conditions when a binary simple condition is defined is to capture both an event state change and the temporal interval during which an event remains in the same state.

An *action role* has the following attributes:

- *id*, which must be unique among the connector set of roles;
- *event type*, which may be presentation, attribution, etc.;
- *action type*, which specifies the type of action to be executed on a participant playing this role. Actions cause transitions on the event state machine (see Figure 4) of participants playing this role. Action type examples are given in Table 1;
- *repeat*, which specifies the number of times the action must be repeated. Its default value is 0;
- *wait time between repetitions*, which defines a time that must be waited before the next repetition. Its default value is 0, and it is only valid if the repeat attribute is greater than 0;
- *value*, necessary and valid only if the event type involves an attribution operation;
- *wait time*, which defines a time that must be waited before the action is executed for the first time. Its default value is 0;
- *cardinality* [*min*, *max*], where $min \geq 1$, $max \geq 1$ and $max \geq min$ (*max* may be ∞). This attribute defines the minimal and maximal number of nodes that may play this action role in an instance of the connector. Its default value is [1,1].

In a causal relation, the connector *glue* defines an expression involving condition roles and another expression involving action roles. When the condition expression is satisfied, the action expression must be fired. In a constraint relation, the connector *glue* specifies the constraint to be satisfied relating condition roles.

The *glue condition expression* of a causal hypermedia connector consists of a logical expression involving condition roles and based on the operators \wedge (and), \vee (or), \neg (not). Besides these operators, there is another operator \oplus (*delay*), which can be applied to any expression operand. The delay operator applied to a condition *C* is denoted as $C \oplus [t_1, t_2]$, where $t_1, t_2 \in \mathfrak{R}$ and $0 \leq t_1 \leq t_2$. Given that a condition *C* is true at instant *t*, a condition *C'*, defined as $C \oplus [t_1, t_2]$, is true at the interval $[t+t_1, t+t_2]$. For example, in order to apply a two-second delay to a compound condition expression involving two roles, the condition expression would be $((role1 \dot{U} role2) \dot{A} [2,2])$.

Table 1. Action type examples that can be applied over presentation and attribution events

Event Type	Action	Description
Presentation	<i>Prepare(E)</i>	If event <i>E</i> is in the sleeping state, it goes to the preparing state; otherwise, there is no transition.
Presentation	<i>Start(E, n)</i>	If event <i>E</i> is in the sleeping, preparing, prepared or paused state, it goes to the occurring state through the state machine (see Figure 4), starting its presentation from the beginning. Otherwise, nothing happens. This action has an optional parameter <i>n</i> to initialize the <i>repeat</i> attribute of <i>E</i> .
Presentation	<i>Stop(E)</i>	If event <i>E</i> is in the occurring or paused state, it goes to the prepared state; otherwise, there is no transition. This action increases by one unit the <i>occurrences</i> attribute of <i>E</i> and decreases by one unit the <i>repeat</i> attribute of <i>E</i> . If the value of <i>repeat</i> is still greater than zero after being decreased, a new presentation of <i>E</i> is automatically started.
Presentation	<i>Pause(E)</i>	If event <i>E</i> is in the occurring state, it goes to the paused state, otherwise, nothing happens.
Presentation	<i>Resume(E)</i>	If event <i>E</i> is in the paused state, it goes back to the occurring state, otherwise, nothing happens. The presentation of <i>E</i> is resumed from the last information unit presented before the event has been paused.
Presentation	<i>Abort(E)</i>	If event <i>E</i> is in the occurring or paused state, it goes to the aborted state and immediately after it goes to the prepared state. The <i>occurrences</i> attribute of <i>E</i> is not increased and the <i>repeat</i> attribute is set to zero. If <i>E</i> is in the preparing state it goes to the sleeping state. If <i>E</i> is in any other state, nothing happens.
Attribution	<i>AbsoluteAssign (E, i)</i>	Assigns value <i>i</i> to the attribute value
Attribution	<i>RelativeAssign (E, i)</i>	Increments the attribute value by <i>i</i>

Condition roles that may be played by more than one hypermedia node ($max > 1$) must have an expression qualifier in the glue condition expression, having the possible following semantics:

- ALL – all conditions must be true;
- ANY – at least one of the conditions must be true.

The *glue action expression* of a causal hypermedia connector is defined by an expression based on the operators | (parallel) and \rightarrow (sequential) involving action roles, specifying the execution order of each action of the expression.

Action roles that may be played by more than one hypermedia node ($max > 1$) must have an expression qualifier in the glue action expression, having the possible following semantics:

- ALL – all actions must be executed;
- ONE – just one of the actions must be executed.

Let us take an example to illustrate a complete causal connector definition. Suppose that a document represented by composite node *COMP2* in Figure 5 uses document *COMP1* already shown in Figure 3. *COMP2* defines link *l3*, which uses causal hypermedia connector *CON2*, also presented in Figure 5. The complete link definition will be discussed in Section 4.1. At first, let us pay attention to its connector.

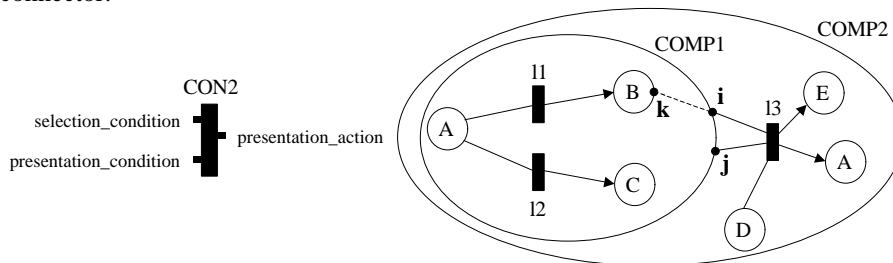


Figure 5. Example of hypermedia document using an n-ary causal connector

Connector *CON2* specifies an n-ary hypermedia relation, where a selection in a node anchor, described by the *selection_condition* role, happening during the second presentation of another node anchor, described by the *presentation_condition* role, fires the presentation of a third node anchor, described by the *presentation_action* role. The fired presentation must occur 2 seconds later and must be repeated twice with one-second delay between the repetitions. The definition of hypermedia connector *CON2* is given by:

- Set of roles:
 1. Condition role:
 - $id \Rightarrow selection_condition$
 - $event\ type \Rightarrow selection$
 - $logical\ expression \Rightarrow (state = occurring, state = prepared)$
 - $cardinality \Rightarrow [1,1]$
 2. Condition role:
 - $id \Rightarrow presentation_condition$
 - $event\ type \Rightarrow presentation$
 - $logical\ expression \Rightarrow ((state = occurring, state = occurring) \hat{U}(true, occurrences = 2))$
 - $cardinality \Rightarrow [1, \forall]$

3. Action role:
 - id => *presentation_action*
 - event type => *presentation*
 - action type => *start*
 - repeat => 2
 - wait time between repetitions => 1
 - wait time => 2
 - cardinality => [1, ∞]
- Glue:
 1. Condition expression => ((*selection_condition*) \hat{U}
(*ALL(presentation_condition)*))
 2. Action expression => *ALL(presentation_action)*

4.1 Hypermedia Links

A *link* uses a hypermedia connector, specifying the nodes that will play each connector role. A link entity has the following attributes:

- hypermedia connector;
- *set of binds*, where each bind defines the node that will play a specific role of the hypermedia connector used.

In a link, each role of its hypermedia connector must have at least the cardinality *min* value of associated binds and at most the cardinality *max* value.

A *bind* has the following attributes:

- node *N*;
- interface point of *N*¹;
- presentation specification of node *N* (optional);
- hypermedia connector;
- role of the hypermedia connector used.

The presentation specification of a node may be specified by a link touching the node in order to tailor the presentation according to the navigation made, for example, associating a new style sheet to an HTML node that is a link target. Some HALs offer this facility, such as NCL.

¹ Node interface points touched by links usually are ports or anchors defining presentation or selection events. Ports are used in composite nodes to export internal node anchors, as was explained previously. However, there may be other types of interface points for defining other types of events as, for example, node attributes, in the case of an attribution event. Without loss of generality, this paper will consider the use of ports and anchors as node interface points.

Returning to the example shown in Figure 3, the set of binds of link l_1 is $\{(A, m, CONI, selection_condition), (B, I, CONI, presentation_action)\}$ and the set of binds of link l_2 is $\{(A, n, CONI, selection_condition), (C, I, CONI, presentation_action)\}$, where anchor I represents the whole content of a node.

In HALs where a node may be reused in different composite nodes and a link may traverse composite nodes (see Figure 5), a link endpoint must identify the nested node sequence used to touch the node occurrence.

Some hypermedia models, like NCM in its previous link definition [16, 17], allow links to directly touch nodes contained in different compositions through the specification of an endpoint nested node sequence, as illustrated in Figure 6(a). Although this is an important feature that must be provided by a hypermedia conceptual model, it also prevents compositionality, since links may go inside and outside a composition without notifying it. In order to allow external links to indirectly touch nodes inside a composite node and also provide compositionality, the use of composite node ports is required.

However, ports alone are not enough for having links touch elements inside a composition. Following the meta-model, composite nodes must define mappings relating its ports to its constituent interface points. These mappings are defined in a composite node attribute, called set of maps. A *map* has the following attributes:

- composite node;
- composite node port;
- constituent node;
- constituent node interface point, which may be an anchor or a port.

Each port of a composite node must have one associated map. A link may indirectly touch a constituent node of a composite node through a port and a path of maps, which leads it to the desired node, as illustrated in Figure 6(b).

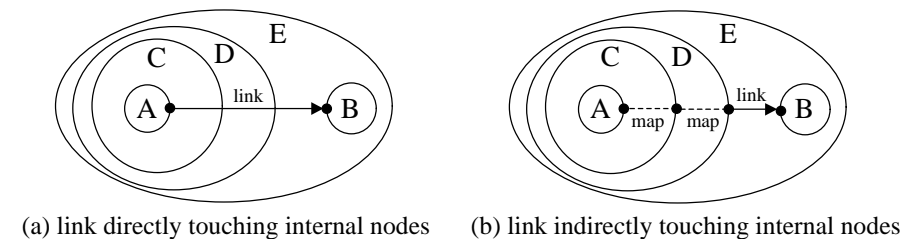


Figure 6. Link definition

Turning back to the example of Figure 5, composite node *COMP1* defines a map between its port *i* and anchor *k* of node *B*, represented by $(COMP1, i, B, k)$, illustrated by a dotted line. Link *l3* specifies that a selection in anchor *k* of node *B*, during the presentation of composite node *COMP1* and during the presentation of content node *D*, fires the presentation of nodes *E* and *A*. The set of binds of *l3* is

$\{(COMP1, i, CON2, selection_condition), (COMP1, j, CON2, presentation_condition), (D, I, CON2, presentation_condition), (E, I, CON2, presentation_action), (A, I, CON2, presentation_action)\}$, where j is an anchor of $COMP1$. The bind related to port i of composite node $COMP1$ associated to map $(COMP1, i, B, k)$ makes anchor k of node B play the $selection_condition$ role of connector $CON2$.

Since a hypermedia connector may be used by different links, if the definition of the connector is changed, all links using that connector must be checked for any inconsistency.

4.2 Composite Hypermedia Connectors

Another consequence of applying the concept of connector to hypermedia is providing composite hypermedia connectors, which is an interesting feature, as we are going to see ahead in the text, not found in any HAL. A *composite hypermedia connector* has the following attributes:

- *set of roles*, that will be associated to constituent connector roles, through the use of maps. The definition of a composite connector role is given by the mapped constituent connector role;
- *glue*, which defines:
 - *set of nodes*;
 - *set of links*;
 - *set of partially defined links*, which may not have binds for all its hypermedia connector roles, but that must have maps for these roles, defined in the following set of maps;
 - *set of maps*, where each map relates a role of the composite connector to a role of a constituent connector occurrence, represented by a partially defined link. Each role of the composite connector must have one associated map.

The graph of nodes, links and partially defined links contained in a composite connector must be connected.

In order to illustrate the definition and the usefulness of composite hypermedia connectors, let us take the example of connector $CON3$, shown in Figure 7. $CON3$ represents a traditional hypermedia relation that requires a user confirmation before performing the navigation to the target node. This is a usual procedure done by commercial web browsers, when the user navigates to a secure web page, for instance. The definition of connector $CON3$ is given by:

- Set of roles:
 1. Condition role: $id \Rightarrow selection_condition$
 2. Action role: $id \Rightarrow presentation_action$
- Glue:

- set of nodes $\Rightarrow \{X\}$
- set of links $\Rightarrow \{\}$
- set of partially defined links $\Rightarrow \{l4, l5\}$
- set of maps $\Rightarrow \{(CON3, selection_condition, l4, selection_condition), (CON3, presentation_action, l5, presentation_action)\}$

The set of partially defined links specifies links $l4$ and $l5$, which use connector $CON1$ presented in Figure 3 and respectively define the following sets of binds $\{(X, I, CON1, presentation_action)\}$ and $\{(X, r, CON1, selection_condition)\}$, where r is anchor of X .

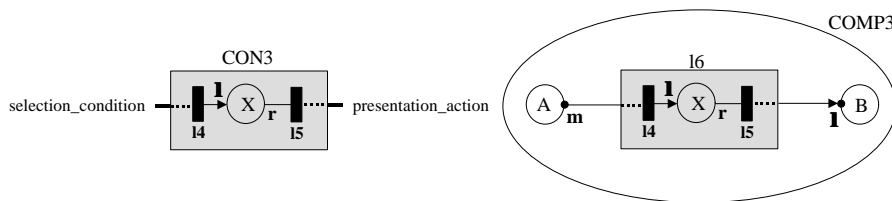


Figure 7. Example of hypermedia document using a composite hypermedia connector

Document $COMP3$ containing link $l6$, which uses connector $CON3$, is also illustrated in Figure 7. The set of binds of $l6$ is $\{(A, m, CON3, selection_condition), (B, I, CON3, presentation_action)\}$. Nodes A and B play the roles of the traditional hypermedia relation, but this navigation involves the presentation of an intermediate node X . When anchor m of node A is selected, link $l4$ fires the presentation of node X . Then, when anchor r of node X is selected, link $l5$ fires the presentation of node B . Considering the example previously given, node X could be the browser message window that is going to warn the user about navigating to a secure web page.

Note that a link using a composite hypermedia connector is defined the same way as a link using a causal or constraint connector. It must reference the hypermedia connector and define the set of binds relating connector roles to node interface points.

5 Issues about the Use of Connectors

The key point for treating spatio-temporal relations as first-class entities is providing the definition of the relation independent of the specification of which nodes will be related.

Some HALs already did this separation when defining a relation. In NCL, for example, the first element of a link, similar to the connector, was called *meeting point* and the second, similar to the set of binds, was called *endpoint set*. However, NCL did not allow a meeting point to be reused in different links and treated both elements of a link as embedded attributes.

Separating the relation specification from a link that will actually use it provides some advantages to hypermedia authoring languages, such as:

- Reusing the same spatio-temporal relation (hypermedia connector) in different links having the same semantics but specifying different interacting nodes.
- Facilitating the definition of high-level relations that a HAL might provide. Now the language does not need to offer only a predefined set of high-level relations to make the authoring process easier, but it can also provide means for creating user-defined ones. Relations that are represented by hypermedia connectors work as link templates and can be used by authors to create links inside their documents. This feature combines more power of expressiveness and more facility of use.
- Treating spatio-temporal hypermedia relations as first-class entities, which can even be composed of other nodes and links.

On the other hand, there is one drawback of specifying links using hypermedia connectors. In the NCM link definition [17], a single unary condition could relate events of different nodes. With connectors, in the logical expression attribute of a condition role, all unary conditions must be related to a single event type (that will be associated to a single node anchor).

For HALs that offer means for specifying spatio-temporal relations through low-level specifications based on events, such as NCL and IMAP [18], the concept of hypermedia connector may be directly applied, providing the advantages highlighted previously. For other HALs that offer a set of high-level relations, such as Madeus [6], the concept of hypermedia connector may also be applied, considering their high-level relations as hypermedia connectors contained in a predefined connector library provided by the language.

The XML Linking Language – XLink [19] also provides support for creating complex links that involve several participants, through its concept of extended link (*extended*-type elements). An extended link has a set of participants and a set of traversal rules. Participants may be local resources (*resource*-type elements) or remote resources (*locator*-type elements) considering the resource where the link is defined. Traversal rules (*arc*-type elements) determine which participants are source or target of the link. Comparing an XLink extended link to links using hypermedia connectors, each traversal rule is represented by a causal hypermedia connector and its set of participants is represented by the set of binds of a link. The same way a bind associates a node and a connector role, each participant of an XLink extended link has a *label* attribute that is going to be used when specifying traversal rules. However, note that a bind specifies the role that a participant will perform in the relation, while a label is just a reference to a participant, without determining its role. This will be specified by traversal rules that reference the label. Analogous to the proposed connectors, the same label may be used by more than one participant in an extended link. An XLink traversal rule relates a source label (*from* attribute) to a target label (*to* attribute), besides specifying when navigation must be made

(*actuate* attribute) and how target node presentation must be done (*show* attribute). Navigation may be triggered by a user selection event (*actuate*="onRequest") or may be done automatically when the starting resource is loaded (*actuate*="onLoad"). The ending resource presentation may be done in a new window (*show*="new"), or may substitute the starting resource presentation (*show*="replace") or may even be embedded in the presentation (*show*="embed"). XLink *actuate* and *show* attribute values respectively represent predefined subsets of condition and action roles of a causal hypermedia connector, involving selection or presentation events. An XLink traversal rule is represented by a hypermedia connector relating just two roles, a condition and an action role, determined by *actuate* and *show* attribute values. The concept of connector proposed in this paper provides much more expressiveness for relation specification, permitting the definition of glue expressions relating any number of roles. Besides that, connector condition and action roles may act on other types of events, besides the usual selection and presentation types. As an example, conditions and actions involving attribution events may be used to specify spatial relations among objects, which is not allowed by XLink. Another limitation of XLink is that it only provides the specification of causal relations. Connectors presented in this work also allow the specification of constraint relations, being able to represent, for example, the semantics of AHM *sync arcs* [4]. Finally, XLink does not permit the definition of traversal rules in an independent way, that is, the definition of a relation without specifying its participants, preventing relation reuse in distinct links.

This work brought features found in the software engineering domain to the hypermedia domain. There are other works in the opposite direction studying the possibility of using document authoring languages in the software engineering domain, such as the work of [11], which discusses the merit of using XML [20] as a meta-language for defining ADLs. Besides that general goal, the authors say that they have developed an XML schema for an ADL called ACME and also a prototype for creating XML representations for architectures defined in ACME.

6 Final Remarks

This work proposed how the concept of connectors could be introduced into hypermedia, bringing some advantages found in ADLs to HALs. Analogously to [13], which discussed why connectors deserve first-class status, this paper discussed why hypermedia synchronization relations, expressed by links, also deserve first-class status. Among its contributions, we can highlight:

- the possibility of reusing a spatio-temporal relation specification in different links with the same behavior;
- the possibility of allowing expert users to specify complex relations, represented by connectors modeling high-level hypermedia relations, which can be easily used by other non-expert authors;

- the possibility of defining composite spatio-temporal relations expressed by links, which is an original feature not found in any hypermedia language or conceptual model;
- the definition of hypermedia composite node ports and set of maps, making a hypermedia model compositional, without losing the feature of anchoring on nodes nested inside composite nodes. Compositionality is desirable in HALs as it facilitates document consistency check.

The concept of hypermedia connector was incorporated into the NCM hypermedia model and into the NCL authoring language, inheriting the advantages discussed. The HyperProp authoring system [17], based on NCM, is currently being updated to provide definition of hypermedia connectors.

A future work, which is already in progress, is a proposal of how the features of hypermedia connectors could be made available for any XML document. The goals of this future work might be compared to the goals of the W3C XLink proposal combined with all the facilities discussed in this paper.

Another future work is studying how the use of ADL architectural styles could be generalized for the hypermedia domain. Styles provide the definition of constraints that every configuration following that style must satisfy. Applying these concepts to hypermedia documents would allow document structures with particular characteristics to be reused in many different documents. These structures could be used as document templates representing well-known types of documents.

References

1. R. Allen. A Formal Approach to Software Architecture, *PhD Thesis*, School of Computer Science, Carnegie Mellon University, May 1997.
2. M.J. Antonacci, D.C. Muchaluat-Saade, R. Rodrigues, L.F.G. Soares. Improving the Expressiveness of XML-Based Hypermedia Authoring Languages, *Proceedings of the International Conference on Multimedia Modeling – MMM'00*, Japan, November 2000.
3. L. Carr, D. Roure, W. Hall, G. Hill. The Distributed Link Service: A Tool for Publishers, Authors and Readers, *Proceedings of the Fourth International World Wide Web Conference*, Boston, USA, 1995.
4. L. Hardman, D.C.A Bulterman, G. Van Rossum. The Amsterdam Hypermedia Model, *Communications of the ACM*, Vol. 37, No. 2, pp. 50-62, February 1994.
5. F. Halasz, M. Schwartz. The Dexter Hypertext Reference Model, *Communications of the ACM*, Vol. 37, No. 2, pp. 30-39, February 1994.
6. M. Jourdan, N. Layaïda, C. Roisin, L. Sabry-Ismail; L. Tardif. Madeus, an Authoring Environment for Interactive Multimedia Documents, *Proceedings of the ACM Multimedia Conference 98*, pp. 267-272, England, September 1998.

7. N. Mehta, N. Medvidovic, S. Phadke. Towards a Taxonomy of Software Connectors, *Proceedings of the 22nd International Conference on Software Engineering*, Limerick, Ireland, June 2000.
8. N. Medvidovic, R. Taylor. A Classification and Comparison Framework for Software Architecture Description Languages, *IEEE Transactions on Software Engineering*, Vol. 26, No. 1, January 2000.
9. D.C. Muchaluat-Saade, L.F.G. Soares. Towards the Convergence between Hypermedia Authoring Languages and Architecture Description Languages, *Proceedings of the ACM Symposium on Document Engineering – SDE'2001*, Atlanta, Georgia, November 2001.
10. M. Pérez-Luque, T. Little. A Temporal Reference Framework for Multimedia Synchronization. *IEEE Journal on Selected Areas in Communications (Special Issue: Synchronization Issues in Multimedia Communication)*, 14(1), January 1996. pp. 36-51.
11. S. Pruitt, D. Stuart, W. Sull, T.W. Cook. The Merit of XML as an Architecture Description Language Meta-Language, *Technical Report of MCC (Microelectronics and Computer Technology Corp.)*, available at <http://www.mcc.com/projects/ssepp/papers/meritofxml.html>, Austin, TX, January 2000.
12. L.M. Rodrigues, R.F. Rodrigues, D.C. Muchaluat-Saade, L.F.G. Soares. Improving SMIL Documents with NCM Facilities, *Proceedings of the VI Multimedia Modeling Conference – MMM'99*, Ottawa, Canada, October 1999.
13. M. Shaw. Procedure Calls are the Assembly Language of Software Interconnections: Connectors Deserve First-Class Status, *Proceedings of the Workshop on Studies of Software Design*, LNCS, Springer-Verlag, 1994.
14. M. Shaw, D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, April 1996.
15. “Synchronized Multimedia Integration Language (SMIL 2.0)” — *W3C Recommendation*, August 2001. available in <http://www.w3.org/TR/smil20>
16. L. Soares, M. Casanova, N. Rodriguez. Nested Composite Nodes and Version Control in an Open Hypermedia System, *International Journal on Information Systems; Special issue on Multimedia Information Systems*, Vol. 20, No. 6, Elsevier Science Ltd. England, pp.501-520, September 1995.
17. L. Soares, R. Rodrigues, D. Muchaluat-Saade. Authoring and Formatting Hypermedia Documents in the HyperProp System, *ACM Multimedia Systems Journal*, Vol. 8, No. 2, pp.118-134, Springer-Verlag, March 2000.
18. M. Vazirgiannis, C. Mourlas. An Object Oriented Model for Interactive Multimedia Applications. *The Computer Journal, British Computer Society*, 36(1), January 1993.
19. “XML Linking Language (XLink) Version 1.0”. *W3C Recommendation*, June 2001. available in <http://www.w3.org/TR/xlink>
20. “Extensible Markup Language (XML) 1.0 (Second Edition)”. *W3C Recommendation*, October 2000. available in <http://www.w3.org/TR/REC-xml>