Design Methodology and Formal Validation of Hypermedia Documents

C.A.S. Santos
saihel@laas.fr

L.F.G. Soares
lfgs@inf.puc-rio.br

G.L. de Souza
guido@dimap.ufrn.br

J-P. Courtiat
courtiat@laas.fr

1 LAAS-CNRS, 7 avenue du Colonel Roche, 31400 Toulouse Cedex, France
2 Dept. Informática, PUC-Rio, R. Marquês S. Vicente 225, 22453-900 - Rio de Janeiro-RJ, Brazil
3 DIMAp, UFRN, Campus Universitário, Lagoa Nova, 59072-970 - Natal-RN, Brazil

ABSTRACT Hypermedia authoring tools usually suffer from a lack of validation capabilities that would make them possible to check a document against temporal inconsistencies. The document design validation methodology proposed in the paper is meant to overcome this problem. The starting point is a document description given in a high-level modeling technique which include hypermedia basic concepts such as nodes (including composite nodes), anchors and links. The high-level document is then translated into a RT-LOTOS formal specification on which simulation, reachability analysis and minimization techniques can be applied for validation purposes. The proposed approach extends an early work to include not only document intrinsic temporal consistency check, but also consistency checking, taking into account the presentation platform.

1. Introduction

In spite of all facilities offered by hypermedia authoring tools, few attentions have been paid to the design of documents free from the inconsistencies that come from temporal constraints applied on their components through relationships among their anchors. In general, inconsistencies come from several combined temporal constraints that depend on the presentation environment. To detect them is a hard task, claiming for a design validation methodology. The benefits of formal methods for multimedia and hypermedia document validation have been exposed in several papers [5,6,8,13].

Early results on temporal consistency checking in multimedia and hypermedia document designs were presented in [5]. There, we introduced a new temporal synchronization model in which basic presentation and constraint objects are join together. That was an ad-hoc proposal based on reverse engineering from the underlying formalism, namely the RT-LOTOS process algebra. That approach, however, has three major drawbacks: First, it is specific. Second, it does not handle the link concept completely. Finally, it does not distinguish a node type from a node instance.

An enhanced approach is discussed in this paper. Its two main contributions are:

1) The proposed methodology includes an automatic translation from a high-level hypermedia model into a RT-LOTOS specification. It relies on those objects that are usually found in hypermedia document descriptions, namely nodes (including composite nodes), anchors and links. The ideas behind the general model discussed in this paper are currently implemented in the HyperProp System [2].

2) Hypermedia document checking against inconsistencies will now be carried out considering the specific platform on which the document will be presented. The platform introduces what can be named “extrinsic constraints”. These constraints correspond to the undesired delays introduced by multimedia presentation devices and also delays introduced by concurrent access to resources. In particular, the paper shows how a basic resource specification library can be created, and how these specifications can be easily combined with the RT-LOTOS specification derived from the high-level hypermedia model.

The paper is organized as follows. Section 2 introduces the high level document model adopted in this work. Section 3 and 4 presents how the Formal Description Technique RT-LOTOS was used to formalize the document model in order verify intrinsic and extrinsic document consistency properties. Section 5 discusses simple examples in order to illustrate the proposed validation methodology and its associated tools for consistency checking. Finally, Section 6 contains the conclusions.
2. Hypermedia documents with temporal constraints

2.1. Hypermedia authoring tools

Hypermedia authoring systems support many tasks, such as: integration of different media types (many times, distributed), definition of temporal and spatial constraints among documents’ components, creation of structured documents, navigation control and presentation monitoring. The authoring tool must be simple to use and sufficiently expressive not to impose a cognitive overhead to authors. Usually, better expressiveness implies on greater probability to generate inconsistent documents. In order to overcome this problem, some systems only permit the use of causal relations between events (start or end of a component presentation) [1,12]. Others, however, consider relative synchronization, where inconsistent presentation specifications can happen [8].

Multimedia and hypermedia applications are real time and performance limited by available resources. For example, video and audio media can need a delay before starting their presentation in order to decompress the information, and the duration and quality of the presentation may be susceptible to network traffic. Moreover, multimedia and hypermedia presentation is dynamic and dependent on users-interactions. For example, if the same information can be presented as a text or an audio (and consequently, with different duration), depending on the user interaction, synchronization constraints will be defined at presentation time. Consistency verification becomes much more difficult in these cases.

Authoring techniques usually postpone design validation to the runtime stage. This is often too late, and the lack of well-established procedure makes the validation process uncertain. A complete formal validation approach is, however, too complex for naive users. They usually prefer a pictorial language ranging from basic timelines to object oriented models [2,7]. That is why the proposed approach has as one of its main requirements to permit authoring documents using the designer’s favorite tool, whose output is then translated into a RT-LOTOS specification for checking against design errors, using partial and, if possible, exhaustive validation techniques. The methodology discussed goes one step beyond, allowing predefined resource specifications, in RT-LOTOS, to be associated with the RT-LOTOS specification derived from the high-level document model. A dynamic validation can thus be applied to the resulting global specification, taking into account a particular platform.

2.2. Hypermedia Documents based in Compositions

The possibility of having relationships among components of a document represents the main characteristic of a hypermedia system. However, it is important to combine this facility with mechanisms to organize the document in order to reduce the so-called “lost in the hyperspace” problem [9]. The structured definition of documents is desirable as it carries built-in concepts of modularity, encapsulation and abstraction. Conceptual Models with composite nodes support such mechanisms, in particular, models that allow nested compositions.

The definition of hypermedia documents in the Nested Context Model (NCM) the conceptual model of the HyperProp System, is based on two familiar concepts, namely nodes and links. Nodes are fragments of information and links interconnect nodes into networks of related nodes. The model goes further and distinguishes two basic classes of nodes, called content and composite nodes, the latter being the central concept of the model [11].

Intuitively, the content node contains data whose internal structure, if any, is application dependent (they are the usual hypermedia nodes). The class of content nodes can be specialized into other classes (text, video, audio, image, etc.), as required by applications.

A composite node C is a node whose content is a collection S of nodes and links such that every base node of every link occurring in S is either C itself or a node occurring in S (the definition of base node of a link is given below - the definitions of link and composition node are indeed mutually recursive). It is considered that an entity E in S is a component of C and that E is contained in C. We also say that a node A is recursively contained in C iff A is contained in C or A is contained in a node recursively contained in C. Note that a component may be included more than once in S. However, in this paper we will assume the links and nodes collection as a set, without loss of generality for our discussion. An important restriction however must be done:
a node cannot be recursively contained in itself. It is worth to note that composite nodes contains nodes and links, generalizing some models that group only nodes in compositions.

As the model allows a node to be recursively contained in different composite nodes and composite nodes to be nested to any depth, it is necessary to introduce the concept of perspective. Intuitively, the perspective of a node identifies through which sequence of nested composite nodes a given node instance is being observed. Formally, a perspective of a node \( N \) is a sequence \( P=(N_0,\ldots,N_i) \), with \( m \geq 1 \), such that \( N_i=N \), \( N_{i+1} \) is a composite node, \( N \) is contained in \( N_{i+1} \), for \( i \in [1,m] \) and \( N_0 \) is not contained in any node. Note that there can be several different perspectives for the same node \( N \), if this node is contained in more than one composite node.

Composite nodes, as defined, have several desired properties, such as:

- Composition nesting, that is, compositions that contain other compositions;
- Grouping of the components of a document and the relationships among them independent of their types (synchronization relationships for presentation, selection relationships for usual hyperlink navigation, etc.);
- Composite nodes use as a new type of node, in all senses, that is:
  - That they can be presented\(^1\) — since in a presentation, it is important to exhibit not only the data content of a document, but also its structure specified in the composite node (for example, when accessing a book chapter modeled as a composite node, besides seeing its content, one may want to visualize its section structuring).
  - That different entry points in a composition can be defined, i.e., that in a composition, components may have different presentations, depending on the entry point. For instance, the duration of a composition (duration of its components exhibition) will depend not only on the duration of its components, but also on the associated entry point;
  - That relations among compositions can be defined.
- Inheritance in the composition nesting, in the sense that relations can be defined in a composition \( C \), referencing components recursively contained in \( C \). This mechanism is extremely important in object reusing.
- Composite node presentation to help user navigation through a document — what can require the use of some filtering mechanism to present the document structure, as discussed in [9], in order to lessen the user disorientation problem.

A detailed comparison among NCM and hypermedia models used in Firefly, CMIF, I-HTSPN and others, with respect to temporal and spatial synchronization aspects as well as to the semantic power of their compositions can be found in [10].

A link has three main attributes, the source end point set, the destination or target end point set and the meeting point. The set of source and destination end points will define events. The meeting point will define relationships among events. An event is defined by the presentation of a marked set of information units of a node (presentation event) or by its selection (selection event) or by the changing of an attribute of a node (attribute event).

The end points are tuples of the form \( <(N_0,\ldots,N_i), \alpha, \text{type}> \) such that \( N_i \) 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 \( k > 0 \), \( \alpha \) is a set of information units (an anchor) of \( N_i \) or an attribute identifier (and its value) of \( N_i \), and type specifies the event type associated with \( \alpha \): (selection, presentation or attribution). The node \( N_i \) is called a base node of the link. The meeting point contains one operation, as usual, composed by one condition and one action. Once the condition is satisfied, the associated action is triggered. The conditions evaluate logical expressions over event states and attribute values of nodes specified by the source end points of the link. Actions in a meeting point are operations that must be executed.

\(^1\) Composite node presentation is different from the presentation of its components. Composite node presentation is the exhibition of the structure defined in the composition and not the exhibition of each one of its components.
over the destination end points of the link, or are just to introduce a time delay. Examples of simple actions are:
start, pause, stop, prepare a presentation event E; wait a time delay; etc.

For each presentation event, one should be able to specify how and with which tool the associated node (data object) will be presented. In HyperProp, descriptor objects contain these specifications. The independence between descriptors and data objects will permit better reuse of objects. Using distinct descriptors, one can define different presentations for the same data object. For example, a text media segment can be presented as text, using descriptor D₁, or it can be synthesized as audio using descriptor D₂. The aggregation of a data object and a descriptor in order to present a component is called representation object.

The descriptor can be specified in composite nodes, in links, in nodes itself or in their classes. When presenting a node, the descriptor explicitly defined on-the-fly by the end user bypasses the descriptors defined during the authoring phase. These in turn have the following precedence order: first, that defined in the composite node that contains the node, if it is the case; second, that defined in the link used to reach the node; third, that defined within the node; and finally the default descriptor defined in the node’s class.

3. Formal specification

The purpose here is not to define a new formalism for describing the temporal structure of hypermedia documents, but instead, to formalize basic and high level hypermedia objects by performing a mapping of these objects onto a general-purpose formal description technique (FDT in short). The FDT is therefore completely hidden to authors. Having an FDT document specification, general-purpose validation techniques (like reachability analysis, model-checking) can be applied for analyzing application oriented properties of the design, in particular consistency properties as identified in [6].

How to perform such mapping is the question. The selected method should present some unique features in order to be useful as well as efficient:

- It must have a formal semantics (and not only an intuitive semantics)
- It should emphasize composition, in the sense that complex document structures should be expressed not by ad-hoc constructions
- It should be able to express non-deterministic behaviors, in particular interactions with the external environment (i.e., the users)
- It should have the ability to express complex time-constrained behaviors
- Finally, it should be mature enough, and software tools for automating the validation procedure should be available.

Process algebra meets the first three requirements and LOTOS (Language of Temporal Ordering Specification) becomes a strong candidate for being an international standard. Since standard LOTOS is not able to express time-constrained behaviors, different extension proposals have been made within the international LOTOS community, and are currently being standardized. Among these proposals, we selected RT-LOTOS (Real-Time LOTOS) proposed at LAAS-CNRS [3,5], in particular because the RTL (Real-Time LOTOS Laboratory) software tool is available and operational [4]. The same approach might easily be adapted to the new emerging LOTOS standard (E-LOTOS) once stabilized with an adequate tool support.

3.1. The specification methodology

The approach developed in the paper addresses the formal semantics of the basic objects of a hypermedia document by providing a mapping between these objects and RT-LOTOS processes, and another mapping between object composition rules and RT-LOTOS process parallel compositions. The approach relies on general mapping rules, as well as on the definition of RT-LOTOS process libraries specifying the behavior of reusable basic hypermedia objects, like content nodes and links. The approach is therefore general and may be fully automated within an authoring system.
3.2. Hypermedia objects specification

The basic structure in a hierarchical composition model of documents is the Composite Node. This general structure is described by the `compositeNode` RT-LOTOS process definition as presented in Figure 1. The following comments may facilitate the specification understanding:

- **Process `compositeNode`** has a formal parameter characterizing the node ID. It features four external gates: gates `start` and `end` are associated with the start and end of the node presentation; gates `user` and `trigger` are related to the capture of user interactions and with the trigger of an anchor, respectively. All these gates are parameterized in order to identify either the nodeID (for gates `start` and `end`) or the anchorID (for gates `user` and `trigger`);

- When process `compositeNode` is instantiated with some `nodeID` value, the process first synchronizes on gate `start` and receives from its outside environment (in fact from the links which started that composite node instance) a value related to the `version` that will be presented. The `nodeID` and `version` parameters are used to find, in the parameters list of the node, the descriptor list of the nodes included in the composite, the descriptor list of the anchors associated with the composite, and the descriptor list of the links defined within the composite. Using the recursive definition of process `compositeNode`, one may note that, once instantiated, the composite node may be re-instantiated again, with different parameters;

```plaintext
process compositeNode[start,end,user,trigger]
(nodeID:nat, nodeList:nodeDescriptorList, anchorList:anchorDescriptorList, linkList:linkDescriptorList) :=

hide endRequest in
  start 'nodeID?version:nat;
  ( ( nodeBody[user,endRequest][version,nodeList,anchorList,linkList] [> end{0}?nodeID!version; exit )
  || [endRequest,end]] terminationNode[endRequest,end][nodeID, version, nodeList] )
  ||
  compositeNode[start,end,user,trigger] (nodeID, nodeList, anchorList, linkList) ) [] exit

where

process nodeBody[user,endRequest]
  (version:nat, nodeList:nodeDescriptorList, anchorList:anchorDescriptorList, linkList:linkDescriptorList) :=

hide start,end,trigger in
  nodePresentations[start,end,user,trigger] (version,nodeList,anchorList,linkList)
  ![start,end,trigger] (version,linkList)

where

process nodePresentations[start,end,user,trigger] (version:nat, nodeList:nodeDescriptorList, anchorList:anchorDescriptorList, linkList:linkDescriptorList) :=
  nodeStructure[start,end,user,trigger] (version,nodeList,anchorList,linkList)
  ![nodeInteractions[user,trigger] (version,anchorList)]

where

process nodeStructure[start,end,user,trigger] (version:nat,nodeList:nodeDescriptorList, anchorList:anchorDescriptorList, linkList:linkDescriptorList) := ...

endproc

process nodeInteractions[user,trigger] (version:nat, anchorList:anchorDescriptorList) :=
  interactionsCapture[user,trigger] (anchor_1)
  ![... || interactionsCapture[user,trigger] (anchor_n)]

endproc

process nodeConstraints[start,end,trigger] (version:nat, linkList:linkDescriptorList) :=
  link[start,end] (descriptorLink_1)
  ![... || link[trigger,end] (descriptorLink_i)]
  ![... || link(trigger,start] (descriptorLink_n)]

endproc

endproc
endproc

Figure 1 - compositeNode generic process definition
```
• Once synchronization is performed on gate \textit{start}, the behavior of the process is essentially described by process \textit{nodeBody} together with the specification of the termination conditions of the composite node (see process \textit{terminationNode}, not further detailed here);

• Process \textit{nodeBody} describes the internal structure of the composite node; it is made up by the parallel composition of process \textit{nodePresentations} and process \textit{nodeConstraints} with mandatory synchronization on some internal gates;

• Process \textit{nodePresentations} represents the logical structure of the node. It is defined as the parallel composition without synchronization of the processes \textit{nodeStructure} and \textit{nodeInteractions}. Process \textit{nodeStructure} corresponds to the parallel composition of the processes associated with the nodes (either content or composite) nested in the current composite node. Process \textit{nodeInteractions} corresponds to the parallel composition of the processes describing user interactions captured on anchors defined in the current composite node. Anchors are modeled by the process \textit{interactionCapture}, which express all possibilities of user interactions, as consecutive interactions, timeouts, delays, etc;

• Process \textit{nodeConstraints} corresponds to the parallel composition with synchronization of the processes characterizing the \textit{links} defined in the current composite node. Process \textit{link} models all synchronization relationships among the components. It is composed by a pair of source and destination actions, in which the time interval to occurrence of the destination action is limited. Compositions of this process \textit{link} can model a set of logical operators;

As suggested by the formal specification described in Figure 1, we did not try to get the most readable formal RT-LOTOS specification, but instead the most general (with respect to the expressive power of the NCM model) and generic, based on reusable components. The goal was to completely automatize the derivation of the formal specification from an NCM authoring. As a consequence, our specification methodology is based on the identification and formalization of several process libraries characterizing, respectively, basic content nodes, basic links, basic composition of nodes and links and basic termination conditions. Some components of these libraries have been inherited from previous work on a simpler hypermedia synchronization model [6].

3.3. Multimedia platform specification

The modularity and hierarchy of a RT-LOTOS specification permit users to combine the process specifying the document presentation (composite node) with another process modeling an available platform. As all interactions are performed by synchronization at the process gates, it is not necessary to distinguish between a centralized and a distributed platform model. The resulting RT-LOTOS composed specification can be checked, using the same methodology described in the next- section, permitting to verify the behavior of a document when presented over a specific platform.

4. Consistency validation

Using formal methods within the context of a high-level authoring tool brings two main advantages. First, it provides a formal semantics to the high-level model, describing without any ambiguity the behavior of a presentation. Second, it permits to check consistency properties on the formal specification derived from the high-level model, using standard validation techniques developed and implemented for the RT-LOTOS in RTL software tool [4]. The verification consists of interpreting the minimal reachable graph that has been achieved by means of RTL. Our purpose in the verification is to prove that the action corresponding to the end of a document presentation is reachable from the initial state of the associated minimal reachable graph. If this occurs, the document presentation finishes and, by definition [6], is consistent. Each node in this graph represents a reachable state. Each arc corresponds to either an action occurrence (event) or a global time progression (arc labeled by \( t \))

Synchronization constraints expressed when specifying a hypermedia document using a high-level approach imply several complex causal and temporal relationships to be fulfilled together when presenting the document. Depending on how these synchronization constraints relate to each other, there is a potential risk to create inconsistent situations, i.e. situations where different contradictory synchronization requirements cannot handled together. Consequently, the document will be deadlocked at some point during its presentation. In [6],
two different inconsistent situations have been identified, namely the qualitative and the quantitative inconsistencies. An inconsistent situation is qualitative if it does not depend on the particular duration of object presentations and quantitative, otherwise.

In this work, we extended the consistency concept, taking into account the interaction between the object presentations and platform resources. Through this new approach, documents may be verified against two properties, named intrinsic and extrinsic temporal consistency.

A document is intrinsically consistent if no internal synchronization constraints can lead to inconsistent situations when the presentation is performed over an ideal platform. However, even if a document is intrinsically qualitative and quantitative consistent, his presentation can lead to a deadlock situation, because the analysis is only based on the intrinsic duration of objects presentations. Therefore, we define a document as extrinsically consistent if its presentation in a real and resource-limited platform is also consistent. In this case, the verification is performed over the composition of the document specification and the platform, testing properties that are not intrinsic to the document, but depend on the behavior of its presentations over a specific platform.

5. Example

Figure 2 depicts the logical structure of a subset of a hypermedia document used as a running example throughout this section. For space and clarity reasons, the example is kept reasonably simple. It nevertheless suffices for illustrating the potential and benefits of the methodology discussed in previous sections.

![Figure 2 – Presentation of a Hypermedia Document](image)

The composition node C is made up of four content nodes V, A, AT and I. It also included links l3, l4 and l5. The presentation of the composition can be started through links l1 or l2, coming from other parts of the document. Node V contains a video. An audio node A contains a music that can be played in background. Node AT has a content, which can be either exhibited as a text or synthesized as an audio, depending on the start link chosen. Finally, node I contains a still picture. The end of the node I presentation triggers link l6, leading to the presentation of another part of the document.

There are two different possibilities to start this composition presentation. When C starts through link l1, nodes V and AT must start at the same time. Node AT must be presented as an audio during d1 seconds, and node V must be presented as a video during d2 seconds. However, if C starts through link l2, nodes V and A must start at the same time with duration of d3 seconds and d4 seconds, respectively. Therefore, the node presentation depends on the external context, that is, on the navigation that led to presentation of the composite node. Furthermore, C can be presented with background music (navigation coming through link l2) or without any background music (navigation coming through link l1).

Link l3 contains a temporal relation stating that node AT must start its presentation d1 seconds after the start of node A. It further provides a descriptor, which defines if AT will be presented either as an audio (of duration d1) or as a text (of duration d2). Link l4 specifies that the image I should be presented just after the end of V and that its presentation may last indefinitely. Consequently, the end of the presentation of node I must be defined by another constraint. This is achieved by link l5, which specifies that node I should end as soon as the presentation of node AT ends.
Figure 3 presents the two presentation scenarios (in timelines) expected by the author. Figure 3a illustrates what happens when one navigates through link $l_1$, assuming that $d_1 \leq d_2$. Figure 3b shows another scenario when one navigates through link $l_2$, assuming that node $AT$ is presented as a text and that $d_3 + d_4 \leq d_2$. None of the figures present any temporal inconsistency.

5.1. Intrinsic Inconsistency

Let us now assume a navigation through link $l_1$, with node $AT$ being presented as a text instead of an audio, and duration $d_2 \geq d_4$. In this case, one may note that node $I$ will never end (because node $V$ ended before the start of node $I$), and consequently the document presentation will never end either, leading to an undesirable deadlock situation, as illustrated in Figure 4a.

On the other hand, the timeline in Figure 4b is based on the presentation depicted by Figure 3b, with node $AT$ presented as an audio with duration $d_4$. At first inspection, the model is still deadlock free. However, this result is obtained without taking care of the resources availability, so the presentation is intrinsically consistent. Figure 4c depicts the minimal reachable graph for the previous scenarios. The reachability of action $end< C,0>$ may be only proved for the scenario corresponding to Figure 4b. The transitions sequences from state 27 to state 1 show that presentation of $I$ has been started and after this, it is blocked in time. After state 18, only the time action can be performed and so, the action $end< C,0>$ is clearly not reachable.
5.2. Extrinsic Consistency

Now that intrinsic consistency analysis was performed, we can think on checking the consistency of the document C against an available presentation platform. Initially, it will be considered that the platform resources are blocking and non-blocking. For example, parallel presentations of different media on the screen do not create inconsistencies, and need not to be checked. Therefore, a screen is non-blocking. On the other hand, an audio channel is blocking, because only one audio presentation can be performed at a time.

A simple modification in the previous formal model permits also to verify the document consistency against a platform with resource limitations. First, two gates – lock and unlock - are included in the node specification to model interactions between presentations and resources. Second, the process terminationNode is modified to guarantee that all resources are released when the composition finishes.

Taking the previous example, it was assumed that: (1) the presentation was performed over a platform with no resource constraints and (2) media delays for presentations were neglected. Therefore, presentations started as soon as their logical and temporal constraints were satisfied and no resource contention occurred during presentation. However, if resources limitations are considered, even when the intrinsic consistency is guaranteed, a document can be inconsistent in a specific platform. Figure 5b shows this situation: the presentation consistent in an initial analysis (Figure 4b) has a conflict between nodes AT and A for the audio channel, when the platform constrains are considered. In the reachability graph (see Figure 6), the highlighted transition shows that the audio channel is occupied by A when AT starts.

The previous analysis also did not consider the delay introduced by resources (due to a CPU sharing, bandwidth limitation, access delays, etc.) in the presentation. In order to handle this situation, another extension of the
platform model is needed, introducing the resource delays. Consequently, it is now possible to model the elapsed time between the instant that a resource is requested to present a content node and the instant when the presentation effectively starts. Figure 5a shows how the intrinsically consistent presentation of the Figure 4a becomes inconsistent if different delays are introduced before video decompression. It is assumed that there is no audio presentation delay. Resource delays can be adjusted by a simple parameter modification in the RT-LOTOS platform specification. This analysis can be extended to model a platform distributed over a network, if this network was modeled by an RT-LOTOS process. Figure 7 shows the minimal reachability graph produced by the RTL tool when we consider the video decompression delay of 5 seconds. Note that the action end\textless{}C,0\textgreater{} cannot be executed when the node presentation starts through link I1, because the duration of V is modified by a platform delay and it finishes after the end of AT node. The presentation is deadlocked on the presentation of image I.

![Reachability Graph of a Presentation Deadlocked by Platform Delays](image)

### 6. Conclusions

From our work, partially discussed in this paper, the following remarks may be drawn. First, it is extremely easy and flexible to work with high-level models at the authoring stage. Second, it is easier for the author to handle document logical structuring than only document presentation structuring. Hence, models based on compositions allowing every type of relationships among their components become very important.[9] Third, the use of a hidden formal description technique, like RT-LOTOS, brings two important advantages: (i) it provides a formal semantics for high level hypermedia models, permitting better understanding of some critical behaviors that may be expressed by the model (ii) it allows applying partial and, when possible, exhaustive validation techniques to check against document design errors.

The methodology discussed allowed predefined resource specifications, in RT-LOTOS, to be associated with the RT-LOTOS specification derived from the high-level document model, so that a dynamic validation could be applied to the resulting global specification, taking into account a particular platform. Hypermedia document checking against temporal inconsistencies can now be carried out considering the specific platform on which the document will be presented.

### 7. References


