XHMBS: A FORMAL MODEL TO SUPPORT HYPERMEDIA SPECIFICATION

*Fabiano B. Paulo**
*Marcelo Augusto S. Turine***
*Maria Cristina F. de Oliveira**
*Paulo C. Masiero**
*Instituto de Ciências Matemáticas de São Carlos
**Instituto de Física de São Carlos
Universidade de São Paulo
C.P. 668, 13560-970 — São Carlos, SP, Brazil
TEL: (016) 273-9688, FAX: (016) 273-9702
E-mail:[mast, cristina, masiero]@icmsc.sc.usp.br

ABSTRACT
This paper introduces XHMBS (the eXtended Hyperdocument Model Based on Statecharts) to support the formal specification of general hypermedia applications. XHMBS uses a novel formalism called hypercharts as its underlying model for specifying the navigational structure, browsing semantics and synchronization requirements of a hyperdocument. Hypercharts are statecharts extended with additional mechanisms for describing the time sequencing and information synchronization requirements typical of multimedia. The extensions incorporated into hypercharts are based on the major characteristics of some Petri net based multimedia models, and make it an alternative to such models for multimedia and hypermedia specification. XHMBS provides facilities for defining the structure of a hypermedia application in terms of nodes and links and also for describing the temporal behavior of dynamic data streams contained in nodes. The model incorporates presentation and communication channels for describing spatial coordination and distribution of information, and anchor objects for ensuring separation between information structure and content.

KEYWORDS: Multimedia/Hypermedia Modeling, Statecharts, Hypercharts, HMBS, XHMBS, Temporal Synchronization, and Formal Specification.

INTRODUCTION
Hypermedia applications integrate the concepts of hypertext and multimedia into a single model for information organization and retrieval. They inherit from hypertexts the non-linear organization of information, and allow their users to navigate and retrieve information by activating links established within the document. The incorporation of multimedia resources, with the simultaneous integration of different static and dynamic(time dependent) media considerably enriches the application's ability to convey information.

Hypermedia specification may be a considerably complex task. In addition to defining the structure of the application in terms of nodes and links, a central issue is the definition of its temporal behavior. Such a behavior is responsible for enforcing the sequencing relationships amongst the different data streams and the synchronization requirements related to the presentation of dynamic data such as audio and video during browsing. Its specification requires adequate models capable of describing sequencing and synchronization policies. According to Blakowski and Steinmetz [1] and Haindl [6], to be applicable to the design of large and complex hypermedia applications, a model should satisfy some important requirements, providing: (i) mechanisms for describing hierarchical structure and hierarchical levels of synchronization; (ii) mechanisms for expressing incomplete timing specifications; (iii) mechanisms for expressing a media object as a single logical unit, abstracting its content while keeping the ability of expressing temporal relationships that refer to parts of the whole media object; (iv) means of expressing a wide range of synchronization patterns; (v) a formal semantics together with verification techniques for detecting inconsistencies in the specification; and (vi) simple and intuitive modeling concepts for authors.

A number of models have been proposed in the literature aimed at providing support for multimedia specification and development. Many are Petri-Net based models that concentrate mainly on the specification of synchronization requirements, such as OCPN (Object Composition Petri Net) [11] and its derivations XOCPN (Extended Object Composition Petri Net) [21], DTPN (Dynamic Timed Petri Nets) [14], TSPN (Time Stream Petri Net) [5], $TSPN_{UI}$ (Time Stream Petri Net with User Interaction) [4].

In contrast to the wide range of choice when selecting a

model for multimedia specification, few models are actually available for the specification of general hypermedia applications. HTSPN (Hierarchical Time Stream Petri Net) [15], MHPN (Multimedia Hypermedia Petri Net) [20] and MORENA [2] (Multimedia Organization Employing a Network Approach) are examples of Petri net based models that cater for the requirements of general hypermedia applications. Vuong and Pereira Filho [22] propose a general and flexible Dexter based model for hypermedia object presentation called HAS (Hypermedia Authoring System). The NCM (Nested Context Model) [16] is a hypermedia conceptual model that supports nested composite nodes and synchronization aspects. The Trellis model was also been extended to support timing requirements [17, 18]. The model proposed in this paper has some similarities with the timing model adopted by Trellis.

This paper introduces a new model called XHMBS (eXtended HMBS) [13,19] that provides a visual formalism for the specification of general hypermedia systems, using hypercharts as an alternative to Petri nets as its underlying model. Statecharts [8,10] are an extension of the finite states machine formalism which incorporates notions of concurrence, hierarchy and event broadcasting, being also initially aimed at the specification of concurrent systems. The hypercharts formalism, on the other hand, extends statecharts to make it suitable for describing the time sequencing and synchronization requirements typical of multimedia applications.

This paper is structured as follows. First we briefly introduce the hyperchart formalism, presenting an overview of its major characteristics. We then present the formal syntax of XHMBS, describing the main features of the proposed model for specifying the structural organization and the browsing semantics of hyperdocuments. Next, we provide an example on the use of XHMBS for describing a hypermedia application, and discuss the advantages and shortcomings of the model compared to other approaches for hypermedia specification. Finally, some concluding remarks are presented, including new research directions.

HYPERCHARTS

Statecharts extend the classical formalism of finite state machines and state transition diagrams by incorporating the notions of hierarchy, orthogonality (concurrency), a broadcast mechanism for communication between concurrent components, composition, aggregation and refinement of states [8]. They have a formal syntax and semantics associated, thus enabling the specification of behavioral aspects of systems in a clear, yet rigorous, manner, and providing means for formal verification and validation of models. Due to space limitations we do not review the basics of statecharts. They are, nonetheless, very well known and available from many sources [8,9,10].

A hyperchart is a conventional statechart extended with three new sets of notations: timed history, timed transitions and synchronization mechanisms. Additionally, notations for state parameterization and transition abstraction are introduced to allow for the compact description of real-sized applications [12,13]. Shortly, a hyperchart structure (HYP) is a 14-tuple:

HYP=<$\mathbf{S},\rho,\psi,\delta,\gamma,\mathbf{V},\mathbf{C},\mathbf{E},\mathbf{T},\mathbf{Ac},\tau,LSC,T\_hist,T_S$>, where:

- $\mathbf{S}$ is the *set of states*; ρ: $\mathbf{S}\rightarrow 2^{\mathbf{S}}$ is a *hierarchy function* that defines the sub-states of each state (a state *s* is *basic* if $\rho(s) = \phi$); ψ: $\mathbf{S}\rightarrow$\{AND,OR\} is a function that defines the type of each state; δ: $\mathbf{S}\rightarrow 2^{\mathbf{S}}$ is the *default function* that defines the set of initial states contained in a state *s;* γ: $\mathbf{H}\rightarrow\mathbf{S}$ is the *history function*, responsible for mapping history symbols to states so that $\gamma(\mathbf{H}) = \{y\}$ if $y \in \mathbf{S}$ and $\psi(y) = \{OR\}$; $\mathbf{V}$ is the *set of expressions* containing identifiers of logical variables; $\mathbf{C}$ is the *set of conditions*, which may be T (*true*), F (*false*) or logical expressions; $\mathbf{E}$ is the *set of event expressions*, also called *labels*; $\mathbf{T} \subset 2^{\mathbf{S}} \times \mathbf{E}\times 2^{\mathbf{S}}$ is the *set of transitions* and $\mathbf{Ac}$ is the *set of actions*.

The above definitions are common both to statecharts and hypercharts. The semantics of h*ypercharts* follow the operational semantics of statecharts, characterized by a sequence of execution steps, each one defining a valid state configuration. A *Statechart legal configuration* (SC) is defined [9,10] as a maximal orthogonal set of *basic states*, representing a possible set of currently active states of the statechart. A global clock controls the execution steps, and each execution step may consist of several sub-steps at which a single transition is activated at a time.

The syntax of the extensions introduced in hypercharts is described below. Their associated semantics is discussed in the following.

- τ: $\mathbf{T}\rightarrow$\{*true*, *false*\} is a timed history function. $\tau(t)$ = *true* implies that transition t has an associated timed history symbol. τ^*: $\mathbf{T}\rightarrow$\{*true*, *false*\} | $\tau^*(t)$ = *true* implies that transition t has a recursive timed history symbol associated. The graphical representation of the timed history notation is a clock icon placed near transition *t*;
- LSC: $\mathbf{S}\rightarrow$N, where N is the set of natural numbers is a *Local Step Counter* function which acts as an execution step counter for a state *s*, registering the total activation time of *s* in terms of execution steps. It is incremented at each execution step of the statechart as long as the state remains active. The system's global clock is the mechanism which allows the incremental updating of the LSC*s*;
- T_hist: S→N is a timing register function responsible for keeping the value of the LSC of a state *s* at the moment *s* was later disabled;
- $\mathbf{T_S} \subseteq (2^{\mathbf{S}\times\mathbf{L}}) \times 2^{\mathbf{S}\times\mathbf{H}} \times sync \times (\mathbf{S} \cup \varnothing)$, in which *sync* is a synchronization type, is the set of *M:N synchronized transitions.* Such transitions, in addition to having a type that defines a synchronization policy, are labeled with timed events (see discussion below).

Hypercharts provide a layer of notation defined in terms of the statechart semantics, so that any hyperchart may be seen as a syntactical description that may be transformed into a

semantically equivalent statechart. It is shown in [12] that all extensions introduced in hypercharts may be described in terms of conventional statechart semantics, so that it is always possible to generate a statechart that exhibits the behavior of a given hyperchart.

Timed history

The assignment of the timed history symbol – the clock icon – to a transition *t* with a target state *s*, as illustrated in Figure 1, implies that, if *s* has been active in the past, the firing of *t* will recover the value of the timed history register of *s*, in addition to recovering the last configuration of *s* (in terms of its sub-states). The recursive version of the timed history symbol implies that the timed history registers of *s* and all its sub-states, recovered by conventional recursive history, will also be recovered.

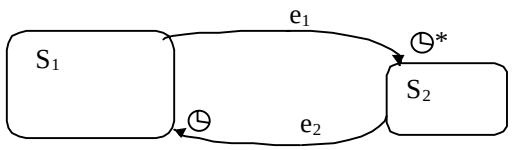


Figure 1: Timed history notation.

To every state $s \in \mathbf{S}$ for which timed history is to be recorded there is an assigned action "*On exit A_Thist(s):=LSC(s)*", ensuring that its timed history register will be updated whenever the system leaves *s*. If there is a transition *t* with (recursive) timed history, whenever *t* is fired, if the states on the target set of *t* have already been active at least once in the past, the operation "*LSC(s):=Thist(s)*" is executed for every state *s* activated by conventional (recursive) history. If the transition is not assigned a timed history symbol, the operation "*LSC(s):=0*" is executed for each state *s* recovered by conventional (recursive) history or activated by default.

Timed history allows activities associated with states to be resumed from the point where they were previously stopped. As soon as the system starts its execution, the operation "*A_LSC(s):=0*" is performed for every state *s*. Additionally, at each execution step *p* the operation "*LSC(s):=LSC(s)+1*" is executed for all the states which were active at the previous step.

Timed transitions

Timed transitions have their firing controlled solely by the time progress during the active time period of their source states. They provide a powerful temporal specification capability useful for specifying time dependent functional behaviors, supporting the specification of multimedia presentation requirements such as delay and jitter. They have assigned timed events of the form "[α,η,β]", and must have cardinality 1:1. Moreover, a state may be the source of only one timed transition.

Figure 2 presents the generic form of a timed transition and its assigned timed event, where *a* denotes an action that may be executed after the firing of the transition, according to the conventional statechart semantics. A timed transition in this example implies that there is a main activity *A* being controlled by state *X* (known as *X's presentation activity*), which generates an associated event *end_X* signaling its termination.

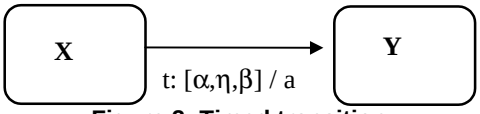


Figure 2: Timed transition.

The notation [α,η,β] imposes two restrictions on the firing of transition *t*. The first is that, even if activity *A* has finished, i.e., *end_X* has occurred, *t* will not fire unless the minimum temporal boundary α has been reached; i.e., unless the application has remained at least α steps at state *X*. If the termination event *end_X* occurs at any relative moment between α and β, *t* will fire immediately. The second restriction is related to the maximum temporal boundary β. If the application remains in state *X* until the relative moment β and event *end_X* does not occur, *t* will fire anyway, aborting the execution of activity *A*.

To model hypermedia applications, it is sufficient to consider that some basic state *X* controls the execution of an activity during its activation time (the throughout option, defined in [8]). Semantically, basic states with a departing timed transition can be decomposed into three exclusive sub-states responsible for controlling the possible firing conditions of the transition, and three variables (*min_X_t,end_X_t,max_X_t*) are used to indicate the exact reason of firing. The termination of the presentation activity is signaled by the internal event e*nd_X* defined for each state *X*. Figure 3 shows the equivalent statechart expansion for the timed transition example depicted by the hyperchart in Figure 2.

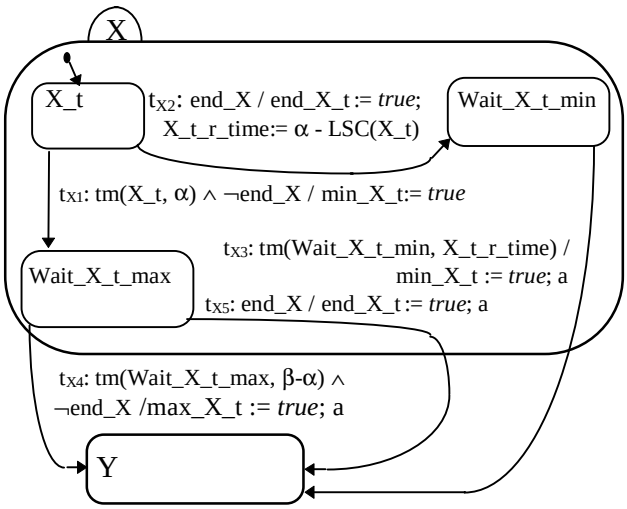


Figure 3: Example of transformation for basic states.

If *X* is not a basic state, an equivalent transformation is obtained by creating a sub-state of *X* that controls the termination of *X*'s presentation activity. This sub-state must be concurrent to any other sub-states of *X*. The formal specification of these transformations may be found in [12].

A special kind of time-out is also defined in hypercharts, denoted by "*tm(s,n)*", in which *s* is an state and *n* is the number of steps along which *s* remained active.

Synchronization mechanisms

In hypercharts, synchronization is specified through *M:N synchronized transitions*, which have five components: source states, source arcs, target states, labels (one assigned to each source arc) and a synchronization type. The type indicates how restrictions imposed by associated timed events will be considered in the generation of a unique temporal restriction for the whole transition. Nine synchronization types are available: five basic ones, namely *strong-or (so)*; *weak-and (wa); master(m)*; *or* and *and;* and four composite types derived from the basic ones, namely *or-master*; *strong-master*; *weak-master* and *and-master*. These are the same adopted in the TSPN and $TSPN_{UI}$ models. An example illustrating the visual representation of a synchronized transition with M = 3 and N = 1 is shown in Figure 4, where T indicates one of the synchronization types. Source arcs must not have labels with associated conditions or actions - the only expression allowed in a source arc's label is a timed event.

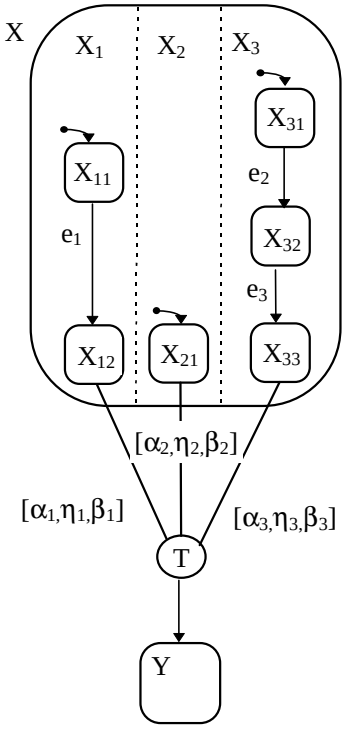
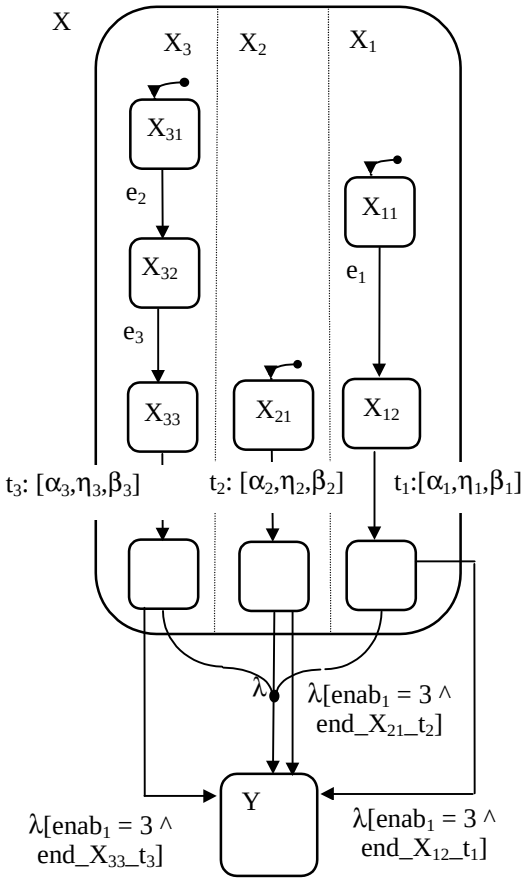


Figure 4: Generic notation for synchronized M:N transitions (with N =1).

An M:N synchronized transition with *M* source states whose respective source arcs are labeled with expressions of the type $[\alpha_i,\eta_i,\beta_i]$ $(1 \le i \le n)$ has its relative firing moment θ in a transition-defined time interval. For example, for an *OR*-type synchronization such interval is given by $\min_i(\alpha_i) \le \theta \le \max_i(\beta_i)$. This is so because the OR synchronization policy considers the temporal correctness of the first of its data streams that successfully finishes its presentation activity. So, the timed transition fires as soon as any state signals the termination of its presentation activity, as long as all of them have already started or, alternatively, as soon as all states have reached their maximum temporal boundary.

The temporal correctness of an activity assigned to a state *s* is guaranteed if the restrictions imposed by the timed transition or by the source arc departing from *s* are satisfied. If a data stream is represented by a set of states mapped to activities that exhibit the data and are related by timed transitions, its temporal correctness is guaranteed if it is guaranteed for every composing state of the data stream. A synchronization scenario is the set of concurrent components of a hyperchart that are part of an M:N synchronized transition. Thus, it always contains two or more data streams to be synchronized at some point. Each of these data streams is represented by an OR component of the hyperchart. Normally, two synchronization points, namely its starting and ending points, define a synchronization scenario.



X: On entry ← $enab_1 := 0$
X_{12}, X_{21}, X_{33} ← $enab_1 := cr(enab_1) + 1$

Figure 5: Transformation for an *or* transition.

The semantic transformation process to derive an equivalent statechart from the hyperchart construction is based on the firing rules defined by the synchronization policy. For example, if the hyperchart in Figure 4 depicted an OR synchronization type, its equivalent statechart representation is depicted in Figure 5. The behavior of the

remaining synchronization types, as well as the formal statechart specifications of the transformations that assign a semantics to the synchronization mechanisms may be found in [12].

A HYPERCHART-BASED HYPERMEDIA MODEL

XHMBS uses the structure and execution semantics of hypercharts to specify both the linked (logical) structure and the browsing semantics of a general hypermedia application. According to the model, a hypermedia application H is a 10-tuple $H = <Hyp, \mathbf{P}, M, ae, L, \mathbf{Pch}, \mathbf{Cch}, V_p, V_a, Ch>$ in which:

- **Hyp** = <**S**, ρ, ψ, δ, γ, **V**, **C**, **E**, **T**, **Ac**, τ, *L_clock*, *T_hist* , $\mathbf{T_S}$> is a hyperchart structure;
- **P** is a *set of pages* corresponding to the atomic information components of the application. Each page $p \in \mathbf{P}$ contains a piece of information that is relevant in some context, being conceptually defined by the triple <c, t, $\mathbf{Anc_p}$>, such that "c" is the information content expressed in any static or dynamic media; "t" represents a title that uniquely identifies the page; $\mathbf{Anc_p}$ defines a collection of anchors associated with the page. Anchors are text strings that once selected by a reader fire corresponding application links. Set **P** also includes a special *null page* with no associated contents, titles or anchors;
- M: $\mathbf{S_S} \leftrightarrow \mathbf{P}$ is a *value function* mapping states into pages. Possible mappings are defined for states in a set $\mathbf{S_S}$: $\{x \in S \mid \psi(x) = OR \vee \rho(x) = \phi\}$, where $\mathbf{S_S}$ is the subset of **S** comprising basic states and OR states. AND states are not mapped into data objects. Null pages may be associated with structural states whose function is not to model information content;
- *ae*: $\mathbf{Anc_P} \rightarrow \mathbf{E}$ defines a function that associates anchors from a page p ($a_p \in \mathbf{Anc_p}$) to hyperchart events that control the firing of transitions. A single event may fire more than one transition;
- L is the hyperdocument browsing level, a natural number defining a visibility level that controls the hierarchy depth when displaying pages during navigation;
- **Pch** is the set of *Pchannels* or presentation channels, abstract devices that support the specification of requirements related to the presentation of information contained in pages;
- **Cch** is the set of *Cchannels* or communication channels defined for distributed applications. These are abstract devices responsible for specifying properties of the interface with the underlying communication sub-system;
- V_P: $\mathbf{P} \rightarrow \mathbf{Pch}$ is the page *visualization function* which associates each page with a presentation channel that is able to interpret it;
- V_{Anc}: $\mathbf{Anc_P} \rightarrow \mathbf{Pch}$ is the anchor *visualization function* which associates each anchor from a page with a presentation channel that is able to interpret it, and
- *Ch:* $\mathbf{Pch} \rightarrow \mathbf{Cch} \cup \varnothing$ is the function which associates a *Cchannel* to each *Pchannel* used by the application. For local applications *Pchannels* may be associated with null C*channels*, that is, *Ch* (*pch*) = $\varnothing$. Similarly to AHM [7], hyperchart channels allow the modeling of presentation and even networking characteristics of the application.

Browsing Semantics

In this section we introduce the browsing semantics associated with XHMBS. According to the model, a hypermedia application consists of a hyperchart representing the logical structure of the application in terms of nodes and links, and its physical structure defined by the "human-consumable" multimedia components and their associated display mechanisms.

The execution semantics of hypercharts/statecharts is used to specify the browsing semantics of the hypermedia application. A set of hyperchart states is mapped into pages of the hypermedia application and the set of events allowed in the hyperchart are associated with anchors representing link sources. The activation of a link from a page results in the triggering of an associated event in its underlying hyperchart model, consequently firing the corresponding transition from a source state (mapped to the page which is the source of the link) to a destination state (mapped to the target page of the link).

The set of pages currently available for navigation in XHMBS is defined by a valid hyperchart state configuration. Users of the application are positioned at an arbitrary number of states at a time, having concurrent access to an arbitrary number of pages. The current hyperchart configuration defines a *user configuration* in the application that determines the set of currently accessible pages at a certain stage during browsing. The presentation of the hypermedia application is made by exhibiting all the pages associated with states in the *user configuration* defined by the current state configuration. To exhibit a page, XHMBS defines a set of presentation objects, namely (Anc_P, Pch, Cch), and a set of mappings related to these objects (*ae*, V_P, V_{Anc}, *Ch*). The associated *pchannels* are invoked for each state within the current user configuration. The *pchannels* are also invoked to interpret and exhibit the anchors within a page - these may be displayed as buttons, menus, marked words, etc. Thus, anchor selection in the application indicates a link activation that triggers a corresponding transition enabled by the associated event in the hyperchart. The firing of the transition results in a new state configuration and the exhibition of the pages associated with its states. For distributed hypermedia applications the corresponding *cchannels* are invoked for each *pchannel*.

Pages associated with ancestral states may also be viewed in conjunction with those associated with states in the current state configuration, a useful facility for allowing users to simultaneously visualize different levels of the hypermedia application hierarchy as mapped into the hyperchart. The hierarchy level to be displayed is specified by the visibility level L, as described below.

If $L = 0$ *then* display all pages p in set D_0,

$$D_0 = \{p \mid x \in S_S \wedge M(x) = p \wedge \rho^0(x) \cap SC \neq \phi\}$$

Generalizing,
If L = *n then* display all pages p in set D_{all}, given by

$D_{all} = U_{i=0...n} D_i$ where

$D_i = \{p \mid x \in S_S \wedge M(x) = p \wedge \rho^i(x) \cap SC \neq \phi\}$

The visibility level supports the simultaneous exploration of different hierarchical levels, providing mechanisms for navigating through the hierarchical structure of the hyperdocument. This type of structured hierarchical navigation is strongly supported by the model, particularly through the use of the *Show-hview* operation. This is defined as a hierarchical view function that shows the pages associated with the states immediately above those in a given hierarchy level L. Formally, let *h* = L + 1 and let *d* be the maximum hyperchart hierarchy level:

Show-hview :

If h ≤ d *then*
display all pages *p* in set D_h,
$D_h = \{p \mid x \in S_S \wedge M(x) = p \wedge \rho^h(x) \cap SC \neq \phi\}$

Execution of the above operation enables three additional operations: UP, DOWN and DISMISS. The only action associated with the latter is the removal of the *Show_hview* results from the display. The *UP* and *DOWN* operations alter the current value of *h*, thus changing the pages displayed as a result of the hierarchical view operation:

UP: *If* h < d *then* h = h+1; *Show_hview*.
DOWN: *If* h > L *then* h = h-1; *Show_hview*.

EXAMPLE

We illustrate the use of the XHMBS model by specifying a hypermedia application that includes a multimedia presentation originally proposed in [4]. The high-level hyperchart specification of the presentation is given in Figure 6. Upon starting a browsing section the reader is presented a text page (actually a menu) represented by its associated state *Menu*. This page provides access to the four contexts of the application: an introduction page (state *Intro*), a photo gallery with a link back to the menu page (state *Photo*); a textual page with information about the author of the hypermedia application (state *About...*); and the multimedia presentation (state *MM_Presentation*).

Basic states *Menu* and *About...* control the activities related to the exhibition of the textual pages. They contain, respectively, an access structure of type index leading to the possible contexts of the application and the author information. From the page associated with the state *About...* there is a link (labeled *menu*) that also leads to the menu page. The link that goes from the menu to the multimedia presentation (state *MM_Presentation*) is a temporal one. Thereby, if after 10 seconds visiting the menu the reader does not activate any link, the multimedia presentation starts.

State *Intro* contains two connection points represented by little arcs labeled with numbers. Connection points are used to represent transition abstraction, according to the notation described in [12]: they indicate that the transitions are going into sub-states of state *Intro* to be specified at another abstraction level, as depicted in Figure 7. This figure presents the decomposition of *Intro* into two orthogonal components to present text and audio concurrently, with an *or*-type synchronization in the end. The introduction page consists of two texts in a sequence, accompanied by a soundtrack. From this page the reader may stop the presentation (event *terminate*) and go to the menu by following a conventional hypermedia application link. The *Photo* state is an orthogonal decomposition containing five sub-states, each controlling the activity responsible for the exhibition of a photograph.

Note that user interaction actions "*skip*", "*change_speed*", "*start*" and "*terminate*" are specified at the highest level (Figure 6), being accessible from any point within the application. The "*terminate*" action is also specified in the context of the multimedia presentation specified by state *MM_Presentation*.

As the specification of jitter for this presentation requires a time granularity level on the order of milliseconds, the speed unit adopted for the hyperchart is 1/ms, that is, an execution step must occur at every millisecond. (Note that interactions "*change_speed*" and "*skip*" in Figure 6 can change this setting through the use of the "*New_G_clock*" operation, that modifies the number of steps occurring in a time unit during the hyperchart execution.) This very requirement imposes the synchronization of audio and video data streams at every third of a second, so that a VS (video sequence), for example, must be described by 15 states, each one representing 1/3 of a second (333 ms). The need for synchronization between audio and video at every third of a second implies that the audio data stream be represented at the same granularity level of the video ones, thereby requiring 30 states for each AS (audio sequence). As the presentation must be specified in both forward and reverse directions, for each state representing a configuration of the forward presentation there must be a corresponding one representing the reverse presentation. Thus, considering only the specification of audio and video components for the presentation, one would have 180 states representing the 6 VSs and another 180 representing the ASs.

Figure 8 depicts the expanded specification of the *MM_Presentation* state. In this Figure one may observe the specification of user interaction actions "*start*", "*terminate*", "*freeze*", "*resume*", "*toggle*" and "*restart*". The *strong-or* M:N synchronized transition (SO) originates from sub-states *Aud_Vid* and *Img_Txt*, and goes into state *MM_idle* to satisfy a requirement stating that presentation of the sixth VS, third AS and second text string must finish simultaneously. Actions expressions assigned to event *tog* ("*toggle*") are responsible for setting the *LSC*s of the target states of the respective transitions, as well for as

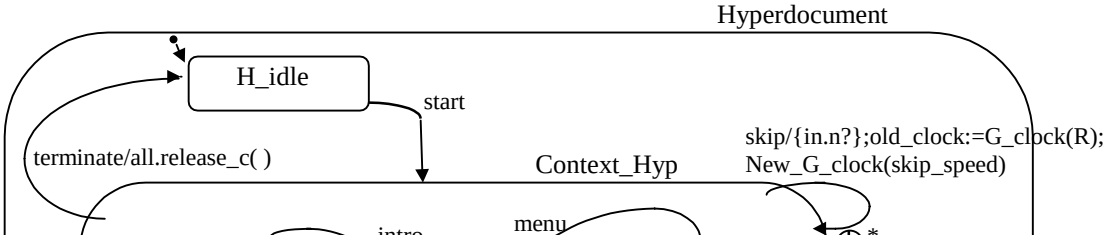


Figure 6: High level hyperchart specification of the hypermedia application.

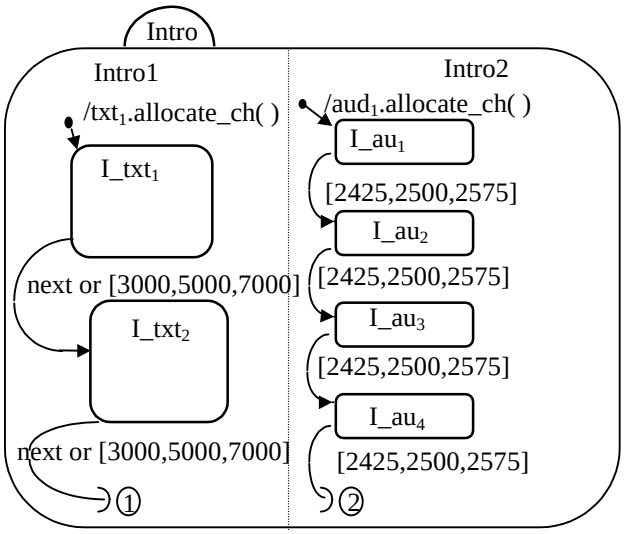


Figure 7: Decomposition state *Intro*.

inverting the presentation direction. The message *all.release_ch( )* was specified as an activity of type "**on entry**" for the state *MM_idle*. This activity ensures that all presentation channels will be released even when state *MM_idle* is activated by the *M:N* synchronized transition, for which action expressions are not allowed. Note also that interaction "*restart*" is allowed to occur in both directions of presentation, and its operation is controlled by variable *d*. As the interaction may happen on the reverse direction (*d=bw*), an *M:N* conventional transition was employed originating from state *MM_Presentation* and going to the terminal sub-state of the presentation (*MM_idle*).

Assuming that the hyperchart model shown in Figure 6 is in a configuration SC_0 = {Menu} and that L=0, let us illustrate possible configurations to be reached during browsing according to the interpretation given to the hyperchart semantics. Such configurations are illustrated as hypothetical layouts of the display window, arbitrarily assuming that pages are mapped into window frames and that available links are indicated by anchors represented as clickable buttons (anchors are labeled with the event associated with the link). The model does not impose any restrictions on the nature of the interface objects associated with pages or links, however.

A schematic layout for the window displaying the application at the initial stage in browsing is presented in figure 9(a). The text page P_{Menu} associated with the *Menu* state in the current state configuration is shown, and the anchors defined for each displayed page are also exhibited. For example, clicking at anchor labeled *General Introduction* activates the event labeled *intro* in the underlying hyperchart, leading to a new state configuration SC_1 = {I_txt_1, I_aud_1} and a new layout for the display, illustrated in figure 9(b). Thus, the presentation contains an aggregate of a text page and an audio page, associated with states I_txt_1 and I_aud_1, respectively, displayed in two separate windows. Through the temporal specifications the presentation may continue on its own. Alternatively, the reader may interact with it by selecting the enabled anchors. For example, one may click at anchor labeled *Next* associated with event *next* of the transition t=(I_txt_1, next or [3000,5000,7000], I_txt_2) (Figure 7), or may select the *Menu* anchor, activating the event *menu* that leads to the page associated with state *Menu*. If the reader selects *Menu* and does not activate any link after 10 seconds, the multimedia presentation starts and the system presents the contents of the pages associated with state "MM_Presentation".

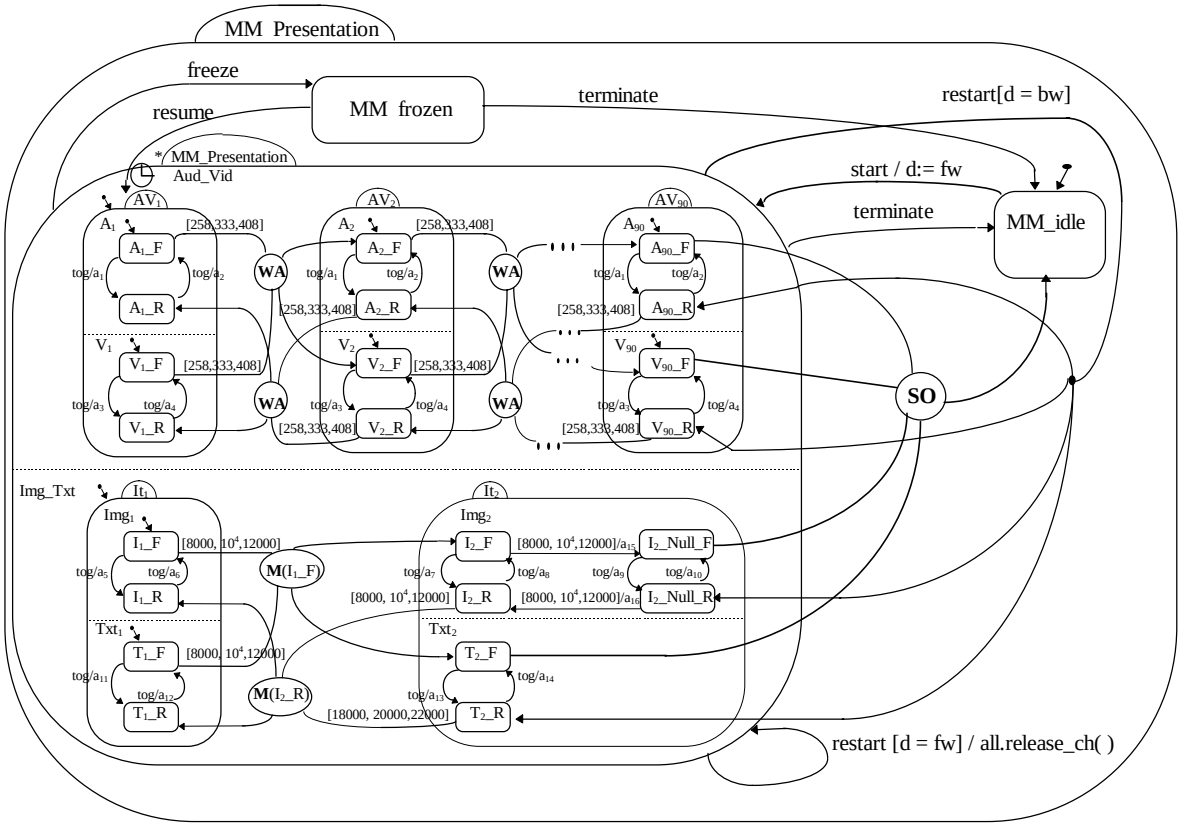
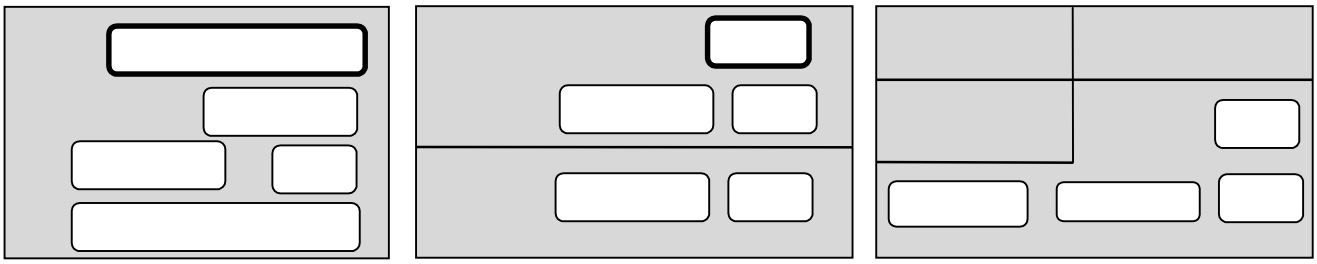


Figure 8: Expanded version of the state *MM_Presentation.*



(a) SC_0 = {Menu}. (b) SC_1 = {I_txt$_1$, I_aud$_1$}. (c) SC_2= {A_1_F, V_1_F, I_1_F, T_1_F}

Figure 9: Possible window layouts displayed during the browsing of the application hypermedia for the given configurations and L=0.

According to the navigational structure presented in figure 8, a text page associated with state *MM_idle* is initially presented. Selecting anchor *Start* associated with event *start*, a new hyperchart configuration is created, SC_2= {A_1_F, V_1_F, I_1_F, T_1_F}. Figure 9(c) presents the four pages associated with the basic states in this configuration. In this case, the reader is not interacting with the multimedia presentation.

DISCUSSION

The Trellis model [17] was originally developed for hypertext applications, and uses Petri nets not only to represent the structure but also to specify the browsing semantics of a hypertext document. Stotts and Furuta [18] extended Trellis incorporating a timing mechanism that modifies the event durations in a document without changing the links in the underlying Petri net.

The MORENA model [2] is a descendant of Trellis targeted at the description and execution of hypermedia applications allowing flexibility and adaptability through message passing. MORENA provides support for dynamic data, such as video or audio, and structured authoring. Composite nodes have their own structure and encapsulate information (such as synchronization specification), defining a hierarchy that can be easily manipulated. Moreover, MORENA provides fine-grained synchronization (a dynamic medium can send synchronization messages at specified moments during its execution), easy prototyping and simulation, and the

ability to reuse logical specifications.

XHMBS also provides mechanism for fine-grained synchronization, and easy prototyping may be supported through the execution of the hyperchart specification. However, a major shortcoming of the Petri net based models is the lack of satisfactory hierarchical structuring facilities, which makes difficult the task of specifying synchronization control across different levels of hierarchical structures. This problem is tackled in XHMBS through the use of hypercharts, which provides inherent support for hierarchical specification of states.

Wang and Wu [20] proposed the MHPN, which links Petri net objects to MHEG (Multimedia and Hypermedia information coding Expert Group) objects to express the complex logic requirements of hypermedia applications, such as flexible browsing and rendering capabilities, and to provide a formal graphical model for the structured authoring of hypermedia. XHMBS might also be extended to integrate MHEG objects.

The NCM model allows authors to specify temporal synchronization relationships among events within hypermedia documents through the usual concepts of anchors and links [16]. XHMBS also has such a characteristic. Both models allow asynchronous specifications, such as user interaction, to be combined with synchronous elements, like video exhibitions, in a single hypermedia document. They also separate the structural organization of the hypermedia document from its presentation and content specification.

CONCLUDING REMARKS

This work presents XHMBS, a novel and effective model that extends the specification power of the HMBS model [19] to attend requirements such as temporal specification and data stream synchronization. This is essentially accomplished by the introduction of hypercharts as the underlying specification formalism to describe both the structural organization and the browsing semantics of a hypermedia application. Other modifications were introduced, such as in the concept of pages, extended to support the association of content, title and anchors. The definition of anchors as objects independent of the page structure and content, and their explicit association with hyperchart events (via the *ae* function) ensures greater flexibility to authors allowing typed links and reuse of events in different contexts.

The model also supports the association of multiple pages to a single hyperchart state, so that multiple versions or perspectives of the same information may be available in a single navigation session. The *Pchannel* (presentation channel) associated with the state is responsible for selecting the page to be exhibited. The Pchannels and Cchannels were inspired in the concept of channels introduced in the Amsterdam Hypermedia Model [7], being more robust abstractions for page interpreters than the readers of the original HMBS model. All the user interaction mechanisms are controlled by the Pchannels, which are classes of objects with attributes and interaction methods. For example, methods *Allocate_ch( )*, *Release_ch( )* and *Init( )* execute, respectively, the allocation and deallocation of a pchannel, and the presentation of a page content submitted to a pchannel.

The XHMBS model successfully addresses the requirements identified by Blakowski and Steinmetz [1] and Haindl [6] for the specification of hypermedia applications. The hierarchical structuring of the application and the structured hierarchical navigation are strongly supported by the model through the *Show-hview* operation. XHMBS separates the structure from the content specification; supports specification at different abstraction and granularity levels; provides mechanisms for specifying navigational contexts and provides suitable synchronization mechanisms for hypermedia [11]. XHMBS relies on a mathematical model (statecharts) with associated semantics and algorithms, yet with an easy and intuitive visual notation associated. It is also an intuitive model for authors with a computer science background, as hypercharts extend the statechart formalism, which is itself an extension of finite-state machines. Modeling based on states and events is well-established within the computer science community. Formal models for hypermedia may provide a standpoint for encouraging application portability among systems. XHMBS is not targeted at the modeling of an application domain, and should be used in connection to a more general design method. Also, it does not provide direct support to version control and for dynamic links. Such issues deserve further investigation, and we believe the model can be extended to deal with them.

We are currently finishing a prototype hyperdocument system called *HySCharts* that supports hyperdocument specification through HMBS model. The *HySCharts* environment includes an authoring and a browsing modes that allow authors to specify the hyperdocument's underlying model based on HMBS and to verify its validity through programmed or interactive execution of the statechart. The prototype is to be extended to support the specification and execution of XHMBS specifications by including facilities for describing multimedia components comprising different media segments.

ACKNOWLEDGMENTS

The authors wish to acknowledge the support of CNPq - The Brazilian National Research Funding Agency - and FAPESP - The State of São Paulo Research Funding Agency.

REFERENCES

1. Blakowski, G. and Steinmetz, R. "A media synchronization survey: reference model, specification, and case studies", *IEEE Journal on Selected Areas in Communications* 14, 1 (January 1996), 5-35.

2. Botafogo, R. and Mossé, D. “The MORENA model for hypermedia authoring and browsing”, in *Proceedings of the International Conference on Multimedia Computing and Systems* (Los Alamitos, CA, USA, 15-18 May 1995), IEEE Computing Society Press, pp.42-49.

3. Buchanan, M.C. and Zellweger, P.T. “Specifying temporal behavior in hypermedia documents”, in *Proceedings of the European Conference on Hypertext*, (Milano, Italy, November 1993), pp.262-271.

4. Cooper, K. “*$TSPN_{UI}$: A Petri net model for specifying user interactions in multimedia presentations*”, MSc Thesis, Canada, The University of British Columbia, 1995.

5. Diaz, M. and Senac, P. “Time stream Petri nets: a model for multimedia streams synchronization”, int *International Conference on Multimedia Modeling*. Singapore, November 1993.

6. Haindl, M. “A new multimedia synchronization model”, *IEEE Journal on Selected Areas in Communications* 14, 1 (January 1996), 73-83.

7. Hardman, L., Bulterman, D.C.A. and Van Rossum, G. “The Amsterdam Hypermedia Model”, Communications of the ACM 37, 2 (February 1994), 50-62.

8. Harel, D. “Statecharts: a visual formalism for complex systems”, *Science of Computer Programming* 8, 1987a, 231-274.

9. Harel, D. “On the formal semantics of Statecharts”, in *Proceedings II IEEE Symposium on Logic in Computer Science* (Ithaca, New York, 1987b), pp.4-64.

10. Harel, D. “On visual formalisms”, *Communications of the ACM* 31, 5 (May 1988), 514-530.

11. Little, T. and Ghafoor, A. “Synchronization and storage for multimedia objects”, *IEEE Journal on Selected Areas in Communications* 8, 3 (April 1990), 413-427.

12. Paulo, F.B. “*Specification of hypermedia applications based on statecharts*”, MSc Dissertation, ICMSC-USP, 1997 (in Portuguese).

13. Paulo, F. B.; Masiero, P. C. and de Oliveira, M. C. F. “Hypercharts: extended Statecharts to support hypermedia specification”, in *Proceedings Third IEEE International Conference on Engineering of Complex Computer Systems, ICECCS* (Como, Italy, September 8-12, 1997), pp.152-161.

14. Prabhakaran, B. and Raghavan, S.V. “Synchronization models for multimedia presentation with user interaction”, in *Proceedings ACM Multimedia 93* (California, USA, 1993), pp.157-166.

15. Sénac, P. “*Contribution to the modeling of multimedia and hypermedia systems*”, PhD Thesis, LAAS, Tolouse, France, June 1996. (in French)

16. Soares, L.F.G.; Casanova, M.A. and 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, 20, 6 (1995),* 501-519.

17. Stotts, P.D. and Furuta, R. “Petri Net based hypertext: document structure with browsing semantics”, *ACM Transactions on Information System* 7,1 (January 1989), 3-29.

18. Stotts, P.D. and Furuta, R. “Dynamic adaptation of hypertext structure”, in *Proceedings Hypertext'91, Third ACM Conference on Hypertext* (San Antonio, Texas, December 15-18, 1991), pp.219-231.

19. Turine, M.A.S.; de Oliveira, M.C.F. and Masiero, P.C. “Designing structured hypertext with HMBS”, in *Proceedings of the VIII International ACM Hypertext Conference* (Southampton, UK, 1997), pp.241-256.

20. Wang, H.K. and Wu, J.L.C. “Interactive hypermedia applications: a model and its implementation”, *Software-Practice and Experience* 25, 9 (September 1995), 1045-1063.

21. Woo, M.; Qazi, N. and Ghafoor, A. “A synchronization framework for communication of pre-orchestrated multimedia information”, *IEEE Network* (January-February 1994).

22. Vuong, S.T. and Pereira Filho, J.G. “HAS: an object oriented model for hypermedia authoring systems”, in *Proceedings of the 1996 Pacific Workshop on Distributed Multimedia Systems, DMS’96* (Hong kong, June 25-28, 1996), pp.71-80.