Synchronization and Storage Models for Multimedia Objects[*]

Thomas DC Little[†]and Arif Ghafoor
Syracuse University, Syracuse, New York 13244-1240

December 1, 1989

Abstract–Multimedia computer systems are required to store, retrieve, and communicate objects comprised of mixed data types including images, text and voice. An important aspect of multimedia systems is the integration of elements retrieved from databases distributed across a network. Integration of object components requires consideration of the temporal characteristics of multimedia elements.

We propose a technique for the formal specification and modeling of multimedia composition with respect to intermedia timing. The proposed model is based on the logic of temporal intervals, and Timed Petri Nets. A strategy is evinced for constructing a database schema to facilitate data storage and retrieval of media elements based on the temporal relationships established by the proposed modeling tool. We present an algorithm which allows the retrieval of media elements from the constructed database in a manner which preserves the temporal requirements of the initial specification. Through the proposed model, the synchronization requirements of complex structures of temporally related objects can be easily specified.

Keywords: Multimedia, modeling, timed Petri nets, distributed databases, synchronization, object modeling.

[*]In *IEEE Journal on Selected Areas in Communications*, Vol. 8, No. 3, April 1990, pp. 413-427.
[†]Now with Boston University, *tdcl@bu.edu*

1 Introduction

Multimedia information systems possess diverse service requirements. These systems must store, retrieve, and communicate complex representations of information under conditions of extreme heterogeneity. Figure 1 indicates the components of a possible distributed multimedia information system and the distribution of tasks on various components for the presentation of services to the end user. The system is comprised of database, network, and workstation elements.

Characteristic of multimedia information systems is the need to compose data of various types and origin for presentation, storage, and communication. Data may be stored locally in singular databases or remotely in multiple heterogeneous database servers. In either case, a mechanism for composing data elements from possibly distributed sources into singular objects is required. This paper addresses the issue of specification of multimedia composition requirements for database storage and retrieval, and the distribution of processing elements for providing a multimedia information service.

Composite representations of data to be integrated in a multimedia application are based upon three types of data: *static*, *dynamic*, and *mixed*. Static types encompass common electronic documents types consisting of text and images arranged in various spatial orientations. Dynamic types include voice annotations, video images, or general animations. Mixed types are formed by uniting both static and dynamic types in a single composite presentation. This type is expected to be the most common in a multimedia environment. The retrieval of data to compose a static object primarily involves spatial organizations of the components comprising the object, as in the case of still images in a textual document. Dynamic object composition requires additional consideration for temporal constraints.

Temporal relationships between the media may be implied, as in the simultaneous acquisition of voice and video, or may be explicitly formulated, as in the case of a multimedia document which possess voice annotated text. In either situation, in order to properly schedule the synchronization of media with vastly different presentation requirements, the characteristics of, and relationships between media must be established.

There has been an effort to standardize the aspects of data integration with respect to spatial types. The Office Document Architecture (ODA) [17] is one such standardization. Our endeavors in this paper are concerned with temporal integration, or *synchronization*. The essence of temporal information within document structure has been discussed in [22]. Reference [10] summarizes the types of temporal presentation control as:

1. sequential,
2. concurrent or simultaneous (synchronous),
3. independent or asynchronous,
4. single medium, and
5. single document.

Three levels of multimedia integration have been proposed in [28]. These are the *physical level*, the *service level*, and the *human interface level*. At the physical level, data from

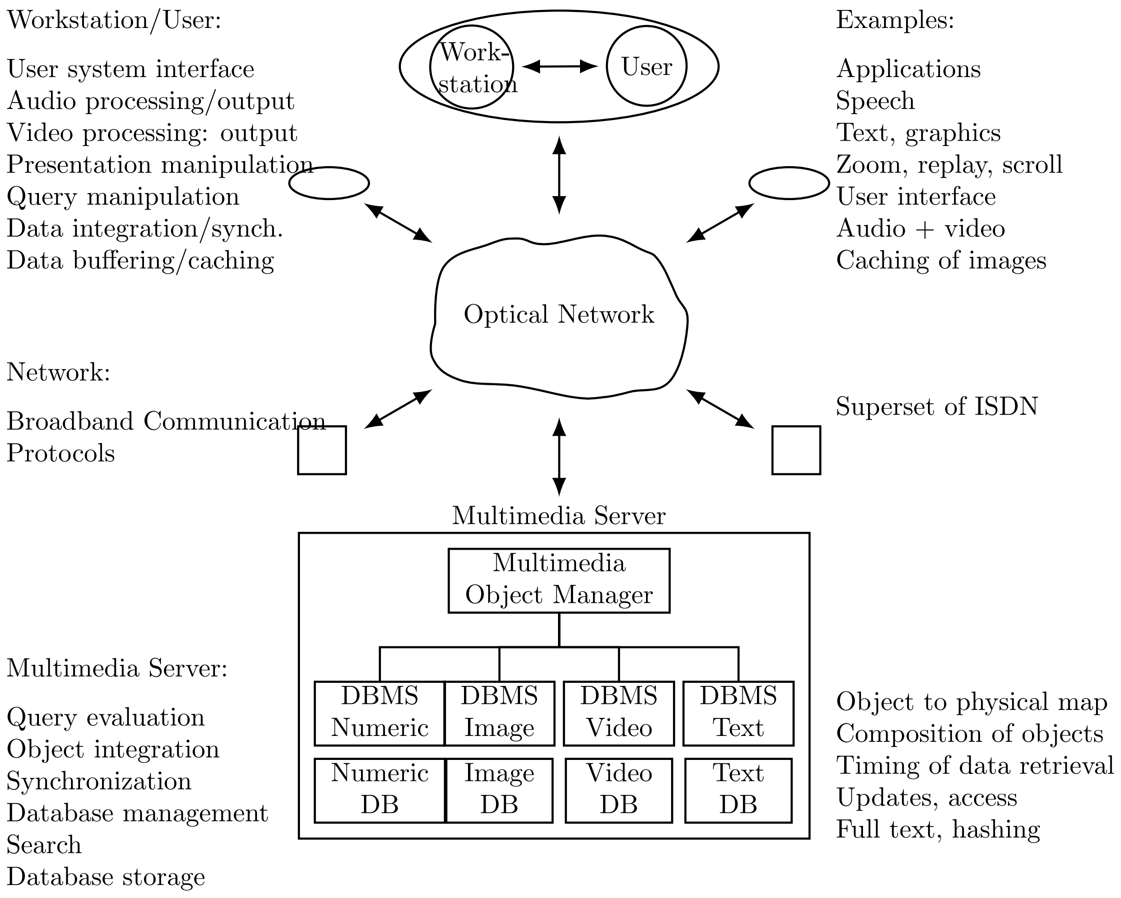


Figure 1: Distributed multimedia information system

different media are multiplexed over single physical connection. The service level is concerned with the interactions between the multimedia application and the various media, and between the elements of the application. The human interface level is deals with the presentation of the different media to the user. The physical and human interface levels are not of primary interest for our discussion. However, the multimedia application requires a coherent presentation of information to the user, which also drives the development of our models. *Presentation* is the term used to denote the activities of delivering the various media to the user. For images and text, presentation implies data display. For audio and video components, presentation indicates a dynamic aural or visual reproduction of the data.

The problem of multimedia synchronization has been addressed by various application environments, however, no formalism has been proposed for general description of multimedia synchronization. The Muse system [14] approaches the problem by grouping display elements of text, video, and graphics into units called *packages*. The packages can be linked together in directed graph structures for the hypermedia paradigm [30], [9]. Coarse-grain synchronization is possible by interconnection of packages. Fine-grain synchronization relies on a binding to a reference timeline. Time is considered a dimension with which other information is arranged by position. Onset and offset times of the presentation of different media can be arranged on a timeline representing their temporal associations. Text and

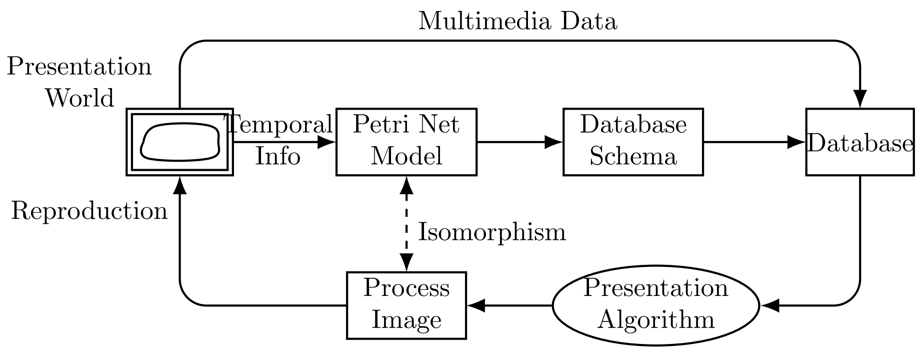


Figure 2: Synchronization and storage modeling for multimedia objects

video information units are linked to a reference timeline, decoupling them from each other, and allowing removal or addition of presentation media without loss of temporal information.

The Command and Control Workstation (CCWS) Project [21] is a multimedia information system designed to support text, vector graphics, bit-mapped images, and digitized speech. Primary functions of the system include real-time conferencing and electronic mail communication among users. Multimedia documents are created with the CCWS using specialized editors. Editing of multimedia objects is compared to viewing a film a frame at a time. Upon creation, temporal relationships are specified between the various media items by icon and relationship manipulation. The temporal relationships are specified by parallel language-like paradigm which can be parsed and interpreted by the presentation component of the system. Presentation of multimedia messages (documents) is facilitated by a recursive-descent parse of the temporal specifications indicated by the editing process. For simultaneous presentation, cooperating processes are created which interact with the operator as required. Each process is assigned a set of frames; one for each element of the current multimedia presentation. Each frame maintains its temporal relation type: sequential or simultaneous, for the current context. Processes are created and destroyed for the duration of a component of the document. These processes appear to have a very loose synchronization with each other for the purpose of coordinating external events such as input from the operator.

Our work in this paper differs from earlier works in several major aspects. We provide a formal specification for synchronization and for the retrieval and presentation for multimedia elements. We propose a explicit representation for synchronization which can be applied to various applications without assuming a particular application environment.

Three contributions to the multimedia data integration problem are indicated. First, we propose a technique for the formal specification and modeling of multimedia composition with respect to intermedia timing. Next, we specify a strategy for constructing a database schema to facilitate data storage and retrieval of media elements based on the temporal relationships established by the proposed modeling tool. Third, we present an algorithm which indicates the retrieval of media elements from the database in a manner which preserves the temporal requirements of the initial specification. Pictorially, these activities are summarized in Figure 2. This figure indicates the relationship between the presentation world, the proposed Petri net modeling technique, a constructed database and schema, and the presentation algorithm.

In Section 2, a discussion of the various data models used is presented, including Petri nets and temporal relations. Database schema and architectural considerations for multimedia synchronization and our model are covered in Section 3. In Section 4, an algorithm for multimedia object retrieval is proposed. A detailed example of the application of our models is provided in Section 5. Section 6 concludes the paper with a discussion of model deficiencies and open issues related to multimedia data composition.

2 Data Models

In this section, we discuss a technique for the synchronization of presentation elements in a multimedia environment without regard to any particular computer or network architecture. We seek an abstract model which characterizes the processes/events associated with presentation of elements with varying display requirements which can be mapped to different multimedia network architectures. The presentation problem requires simultaneous, sequential, and independent display of heterogeneous data. This problem closely resembles that of the execution of sequential and parallel threads in a concurrent computational system, for which numerous approaches exist. Many computer languages support this concept, for example, CSP [13] and Ada [16]. The problem, however, differs in a number of ways.

First, computational systems are generally interested in the solution of problems which desire high throughput, for example, the parallel solution to matrix inversion. This is an important distinction. Multimedia presentation is concerned with the coherent presentation of heterogeneous media to a user; therefore, there exists a bound on the speed of delivery beyond which a user cannot assimilate the information content of the presentation. For computational systems it is always desired to produce a solution in minimum time. A specification model is desired which captures the time dependencies of the presentation of multimedia data and indicates sequential and parallel activities. In terms of our model, the abstract synchronization model is concerned with *presentation*, in contrast to *computation*.

Second, most computational systems rely on precompiled program code for execution. Parallelism is often determined prior to compile-time. Problem solutions are based upon the prespecification of a flow of control in the form of programming language control constructs for each problem encountered. This is contrasted with our desire to present stored data in sequential, parallel, and independent manner with a single, invariable, executable control entity which may determine presentation control flow from the contents of the database. Rather than producing compiled language code for each multimedia presentation, we choose to maintain an equivalent semantic representation in a multimedia database to provide similar functionality. In so doing, changes in the contents of the multimedia database facilitate alterations control flow during multimedia presentation without modification of the single control entity. In summary, distinctions between presentation and computation processing are found in the time dependencies of processing versus display, and the nature of the storage of control flow information.

2.1 Multimedia Process Synchronization Modeling

Consider the presentation of voice and images in a multimedia slide presentation. We would like to represent the interactions between these two media. Several issues arise. How do we describe the temporal relationships and how do we model them for subsequent storage and retrieval? In the general case, a technique to model the real-world in a succinct way is sought. We choose Petri nets as our modeling tool for their ability to specify real-time process interaction based on hard, interprocess timing relationships, as required for multimedia presentation. Formal languages such as CSP give us concurrency but not real-time timing specification as the Petri net does. Additionally, Petri nets are amenable to analyses including Markov process modeling.

Description of the temporal relationships is facilitated based on the concept of temporal intervals, which will be discussed in Section 4 for the purpose of building a database schema.

2.2 Petri Nets

The presentation of multiple media is inherently a task which assumes parallel and sequential activities. In this case, the parallelism is in presentation and processing of multiple data streams. We have chosen the Petri net [20], [1], for a representation of the synchronization of multimedia entities. More specifically, we consider the timed [23], [15], [31], and augmented [7] Petri net models. These models are chosen for their desirable attributes of representation of concurrent and asynchronous events.

The Petri net is defined as a bipartite, directed graph $N = (T, P, A)$ where [15],

$$\begin{aligned} &T = \{t_1, t_2, \ldots, t_n\}, \\ &P = \{p_1, p_2, \ldots, p_m\}, \text{ and} \\ &A\colon \{T \times P\} \cup \{P \times T\} \to I, \quad I = \{1, 2, \ldots\}. \end{aligned}$$

T, P, and A represent a set of *transitions* (bars), a set of *places* (circles), and a set of directed arcs, respectively.

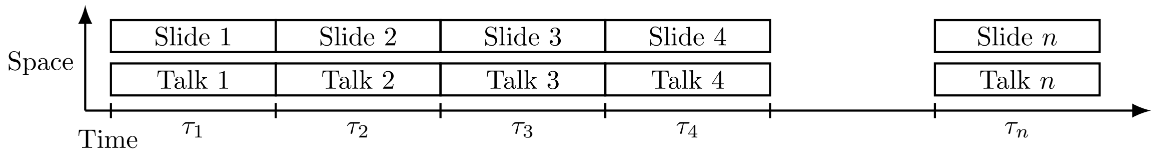


Figure 3: Slide presentation timeline

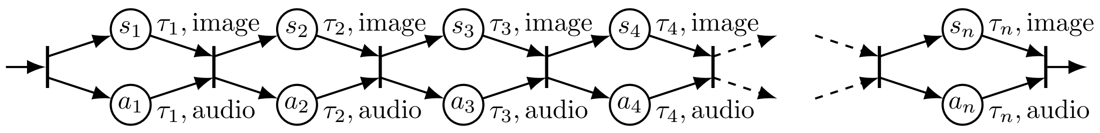


Figure 4: Slide presentation Petri net

A *marked Petri net* $(N_M) = \{T, P, A, M\}$ includes a marking M which assigns *tokens* (dots) to each place in the net:

$$M\colon P \to I, \quad I = \{0, 1, 2, \dots\}.$$

M is a mapping from the set of places to the integers.

For simple Petri nets, the time from enabling a transition to firing is unspecified and indeterminate. Firing of a transition is assumed to be an instantaneous event. To represent the concept of nonzero time expenditure in the Petri net, extensions of the original model are required. A class of enhanced Petri net models have been developed which assign a firing duration to each transition [23], [15], [31]. These models are generally called *timed* Petri net (TPN) models, and map well to Markov performance analysis.

Another TPN model [7] represents processes by places instead of transitions. Nonnegative execution times are assigned to each place in the net. With this scheme the notion of instantaneous firing of transitions is preserved, and the state of the system is always clearly represented during process execution (tokens are at all times in places, not transitions). This "augmented" model has the advantage of compactness of representation. Either process timing scheme can be used for our purposes; however, we choose the more compact of the representations. We supplement the augmented model with resource information associated with TPN models [23], for the purpose of illustrating the use of the multiple media.

Summarizing, we suggest a modification of earlier Petri net models, and call the new model Object Composition Petri Net (OCPN) for distinction from other PN models. The OCPN augments the conventional Petri net model with values of time, as durations, and resource utilization on the places in the net; therefore, an OCPN is defined as $C_{\mathrm{OCPN}} = \{T, P, A, D, R, M\}$ where additionally,

$$\begin{aligned} &D\colon P \to R \text{ and} \\ &R\colon P \to \{r_1, r_2, \dots, r_k\}. \end{aligned}$$

D and R are mappings from the set of places to the real numbers (*durations*), and from the set of places to a set of *resources*, respectively.

Associated with the definition of the Petri net is a set of firing rules governing the semantics of the model. Since we define a transition to occur instantaneously, places rather than transitions have states. The firing rules are summarized as follows.

1. A transition t_i fires immediately when each of its input places contain an *unlocked* token.

2. Upon firing, the transition t_i removes a token from each of its input places and adds a token to each of its output places.

3. After receiving a token, a place p_j remains in the active state for the interval specified by the duration τ_j. During this interval, the token is *locked*. When the place becomes inactive, or upon expiration of the duration τ_j, the token becomes *unlocked*.

For illustration of these concepts, consider the following example.

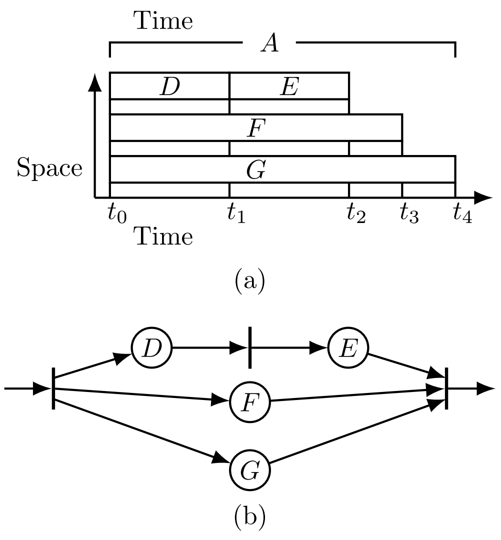


Figure 5: (a) Complex example timeline. (b) Complex example Petri net

Example 1: A multimedia slide presentation is to be represented. The presentation consists of a sequence of synchronized audio and visual elements of varying duration. Assume that there are n slides to present, and corresponding to each slide is a verbal annotation. On a time/resource line, we can represent the presentation as two streams of information which occur concurrently, as in Figure 3.

Time is indicated on the horizontal axis. Resources are described by a dimension on the vertical axis indicating multiple threads of presentation. We call this axis *space*. Using our Petri net representation, these activities are indicated by the Petri net shown in Figure 4.

For each place we have assigned the required presentation resource (device), and the time required to output the presentation data. The transitions in the net indicate points of synchronization, and the places processing. For example, upon completion of the first verbal annotation, place a_1, represented by $talk_1$, unlocks its token. Because elements of the image (slide) and audio data streams have the same respective durations, the place s_1, indicated by $slide_1$, unlocks its token synchronously with place a_1. The common transition fires immediately, allowing the next image/audio pair to be presented.

It is possible to represent arbitrarily complex synchronization with this technique. Consider a hypothetical time/resource representation for the synchronization of a hypothetical object hierarchy shown in Figure 5a. Here we indicate the overall presentation of an object A composed of subobjects D, E, F, and G. This example might correspond to the presentation of text (G), voice annotation (F), and two sequential images (D and E). The Petri net representation of this example is shown in Figure 5b.

Various degrees of granularity are possible, for example, full-motion video usually has an associated audio component. If one decomposes the video into individual frames, which are output at 30/s [19], matching audio can be synchronized in segments corresponding to frames in a Petri net structure as in Section 2.2. Videodisk technology permits access of individual frames of full-motion video, which is important for building multimedia systems.

So far, we have indicated a methodology for specifying related presentation processes associated with different media. We have considered sequential and parallel relations; two elements of a larger class of relations on temporal intervals, which are defined in the next section. For a general process model that represents all potential multimedia scenarios we draw from the field of time modeling.

2.3 Temporal Intervals

Temporal information has been modeled in information systems at the conceptual level, for example, [5], [3], [27], [24]. Bolour *et al.* [4] present an extensive survey of work on the role of time in information processing, with an emphasis on temporal logic and artificial intelligence. Two principle applications of this research are historical database maintenance and inference generation in temporal systems, both of interest in our study but primarily the latter. Reference [25] indicates a taxonomy of time for use in databases, and demonstrates a pictorial representation of time in them. Query languages have been developed for reference to temporal information, for example, TQel [26] and Time-by-Example [29]. Reasoning about time is the topic of a great deal of work. Some recent work includes [12], [18], and [2].

One of the central themes in temporal information modeling has been how to represent time; instantaneously, or non-instantaneously. The merits of each scheme are reflected in the logical systems proposed for each. We make no attempt to further these systems. One noninstantaneous representation of time uses *temporal intervals* [12].

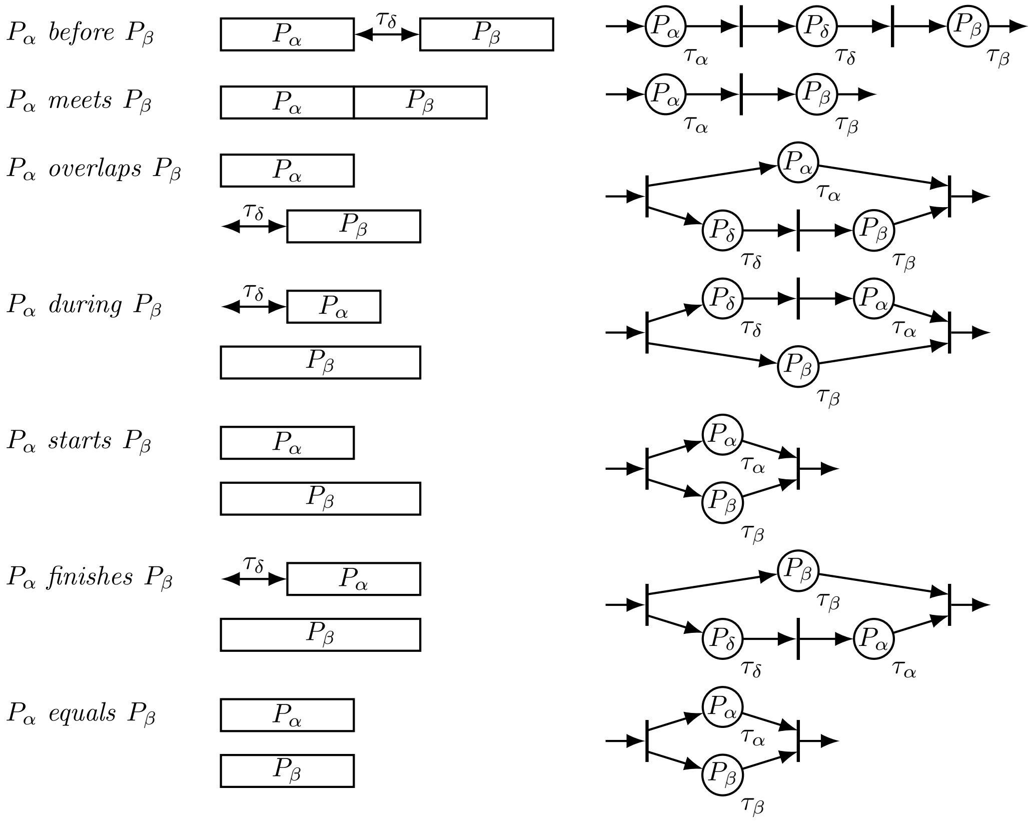
A temporal interval is characterized as a nonzero duration of time in any set of units, for example, “a week” or “100 ms.” This is contrasted with a time instant which is a point structure with a zero length duration; “12:00 AM.” Formally, we can define intervals as follows [3]:

Definition: Let $[\,S, \leq\,]$ be a partially ordered set, and let a, b be any two elements of S such that $a \leq b$. The set $\{x \mid a \leq x \leq b\}$ is called an *interval* of S denoted by $[a, b]$. Any interval $[a, b]$ of S has the following properties:

1. $[a, b] = [c, d] \Leftrightarrow a = c$ and $b = d$
2. if $c, d \in [a, b]$ and $e \in S$ and $c \leq e \leq d$ then $e \in [a, b]$
3. $\#([a, b]) > 1$.

Hamblin presents a logic of intervals [12] which is very useful in the development of a synchronization scheme. Given any two intervals, there are thirteen distinct ways in which they can be related. These relations indicate how two intervals relate in time; whether they overlap, abut, precede, etc. Using the representation of [2], these relations are indicated graphically by a timeline representation of Figure 6a. We only show seven of the thirteen relations since the remainder are inverse relations. For example, *after* is the inverse relation of *before*, or equivalently, *before*$^{-1}$ is the inverse relation of *before*. For inverse relations, given any two intervals, it is possible to represent their relation by using the noninverse relations only by exchanging the interval labels. Note that the equality relation has no inverse.

In Figure 6a we show intervals as processes P_α and P_β with durations τ_α and τ_β, respectively. One axis indicates the time dimension, and the other *space*, or resource utilization, as previously discussed. We define an *atomic* process to be one which cannot be decomposed into subprocesses, as in the case of the presentation of a single frame of a motion picture. The following theorem relates temporal intervals to object composition Petri nets.



P_α before P_β
P_α meets P_β
P_α overlaps P_β
P_α during P_β
P_α starts P_β
P_α finishes P_β
P_α equals P_β
P_α
P_β
P_δ
τ_α
τ_β
τ_δ

(a) (b)

Figure 6: (a) Temporal relations. (b) Corresponding Petri nets

Theorem 1 *Given any two atomic processes specified by temporal intervals, there exists a Petri net (OCPN) representation for their relationship in time.*

We show perfect induction by constructing a Petri net representation for each temporal interval by assigning a place for each process with associated resource and duration. A delay process and place is introduced depending on the type of temporal relationship. Let P_α, and P_β be atomic processes with finite, nontrivial (nonzero) temporal intervals τ_α, and τ_β, respectively. Let τ_δ be the finite delay duration specific to any temporal relation T_R. We construct a Petri net for each relation, indicated in Figure 6b.

For the *meets*, *starts*, and *equals* relations, $\tau_\delta = 0$. For the *overlaps*, *during*, and *starts* relations, the model does not explicitly indicate the delay of the process which completes first. Since both places (processes) lead to the same transition and the Petri net requires that tokens from each be unlocked prior to the execution of the succeeding process, the Petri net ensures synchronization for these cases. The remaining inverse temporal relations possess Petri net representations by a relabeling of the operands and will not be considered here.

For any two atomic processes and their temporal relation there is a corresponding Petri net (OCPN) model. The converse is true as well; for any OCPN model a corresponding temporal relation can be uniquely identified. Consider the Petri net of Figure 7. This unified model serves to represent any temporal relation, not in the most compact form, but reducing to

Table 1: Temporal parameters of unified model

Relation	Parameter Relationships		Overall Duration
P_α *before* P_β		$\tau_\delta \neq 0$	$\tau_{\mathrm{TR}} = \tau_\alpha + \tau_\beta + \tau_\delta$
P_α *meets* P_β		$\tau_\delta = 0$	$\tau_{\mathrm{TR}} \geq \tau_\alpha + \tau_\beta$
P_α *overlaps* P_β	$\tau_\alpha < \tau_\beta + \tau_\delta$,	$\tau_\delta \neq 0$	$\tau_{\mathrm{TR}} = \tau_\beta + \tau_\delta$
P_α *during*$^{-1}$ P_β	$\tau_\alpha > \tau_\beta + \tau_\delta$,	$\tau_\delta \neq 0$	$\tau_{\mathrm{TR}} = \tau_\alpha$
P_α *starts* P_β	$\tau_\alpha < \tau_\beta$,	$\tau_\delta = 0$	$\tau_{\mathrm{TR}} = \tau_\beta$
P_α *finishes*$^{-1}$ P_β	$\tau_\alpha = \tau_\beta + \tau_\delta$,	$\tau_\delta \neq 0$	$\tau_{\mathrm{TR}} = \tau_\alpha$
P_α *equals* P_β	$\tau_\alpha = \tau_\beta$,	$\tau_\delta = 0$	$\tau_{\mathrm{TR}} = \tau_\alpha$

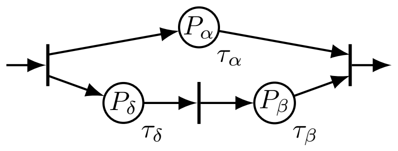


Figure 7: Unified OCPN model

the forms indicated in Figure 6b. It will serve to demonstrate the uniqueness of the OCPN model for each temporal relation.

Let P_α, and P_β be processes with temporal intervals τ_α and τ_β, respectively, temporal relation T_R, and delay τ_δ. By inspection, we determine the distinction between the overall duration, τ_{TR}, of pairs of processes indicated by sequential and parallel relations. For the sequential cases, $\tau_{\mathrm{TR}} \geq \tau_\alpha + \tau_\beta$. For the parallel cases, $\tau_{\mathrm{TR}} < \tau_\alpha + \tau_\beta$. Table 1 summarizes the temporal parameters for each relation specific to the unified model of Figure 7.

For the sequential cases we can distinguish *before* from *meets* by the delay constant only. The parallel cases are identifiable by the characteristics of τ_α and τ_δ. Therefore, given an OCPN model of a pair of processes consisting of three timing parameters and associated Petri net structure (identifying parallel or sequential cases) the isomorphic temporal relationship can be identified.

We directly map the atomic presentation processes to Petri net places, and associate additional delay processes to facilitate the proper timing relationships. By using the hierarchical modeling property of the Petri net we can replace *subnets* of a Petri net by equivalent abstract places [20]. Figure 8 indicates an example of this operation.

Let P_γ represent the process model formed by relating the processes P_α, and P_β in time, where both P_α and P_β are atomic. For these two processes and their temporal relation we can build a Petri net. Since each process is represented by a subnet in the net, we can say that subnet SPN_γ can be replaced by a place p_γ. Thus, we allow a subnet to replace an equivalent abstract place, and vice versa. The following lemma characterizes the replacement of subnets by equivalent places:

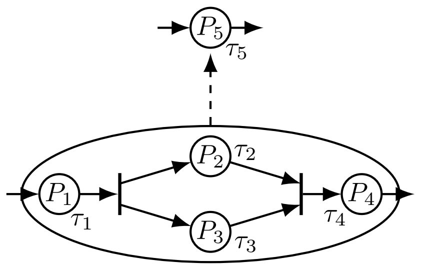


Figure 8: Example of subnet replacement

Lemma 1 *The duration, $\tau_{\gamma+1}$, of process $P_{\gamma+1}$ can be uniquely determined by the duration of subprocesses $P_{\alpha,\gamma}(\tau_{\alpha,\gamma})$, $P_{\beta,\gamma}(\tau_{\beta,\gamma})$, $\tau_{\delta,\gamma}$, and the temporal relation between them; $T_{R\gamma}$.*

For the sequential cases of $T_{R\gamma+1}$; *before* and *meets*, duration $\tau_{\alpha,\gamma+1} = \tau_{\alpha,\gamma} + \tau_{\beta,\gamma} + \tau_{\delta,\gamma}$. For *overlaps*, $\tau_{\alpha,\gamma+1} = \tau_{\beta,\gamma} + \tau_{\delta,\gamma}$. For the remaining parallel cases, *during*, *starts*, *finishes*, and *equals*, $\tau_{\alpha,\gamma+1} = \max(\tau_{\alpha,\gamma}, \tau_{\beta,\gamma})$. All are by inspection of the definition of the thirteen temporal relations.

Using subnet replacement, and Lemma 1, we can state the following theorem for the construction of process models:

Theorem 2 *An arbitrarily complex process model composed of temporal relations can be constructed with Petri nets by choosing pairwise, temporal relationships between process entities.*

Given any two atomic processes P_α, P_β, and their associated timing relationships, we can form, by Theorem 1, a Petri net indicating their combined process interaction in time. By virtue of Lemma 1 and the Petri net abstraction process, this subnet can be called SPN_i, where i is assigned uniquely to subnets as they are constructed. That is, i is a monotonically increasing integer with net construction, or, for all i, $i < j$, SPN_i is not a proper superset of SPN_j (intuitively this should be: for all i, $i < j$, SPN_i is a proper subset of SPN_j, but the Petri net can have more than one connected component). The subnet can be replaced by an equivalent place which we call p_i. p_i represents the complex process P_i comprised of subprocesses indicated by the subnet. For some k where $k \geq 3$, let us assume that the complex process represented by P_k is constructed from complex processes P_l, and P_m where l and m are distinct integers less than k. Process P_l, P_m, and their temporal relation T_{Rk} form a Petri net, or subnet SPN_k. Suppose that we have a complex process P_{k+1} which is composed of process P_β and processes from P_k, and that no Petri net representation exists. Since a subnet SPN_k exists for the process model P_k, it can be replaced by an equivalent Petri net place p_k, with temporal interval derivable from its components, by Lemma 1. Since composite process P_k can be represented with a single duration, and process P_β is atomic, a Petri net representation can be selected by Theorem 1 and the relation in time between the two processes, contradicting our assumption that no Petri net exists. Therefore, process models represented by temporal relationships of any complexity can be represented by Petri net models.

Using the OCPN models of Theorem 1 and the construction process of Theorem 2, it is clear that no modeling of Petri net *conflicts* is done. Stated differently, places of OCPN models have exactly one incoming and one outgoing arc. Therefore, OCPN's are a form of *marked graph* [7], and possess their characteristics. The following is a discussion of some important properties of OCPN's.

Reachability: OCPN's are deterministic because no conflicts are modeled, transitions are instantaneous, and tokens are remain at places for known, finite durations. The result is a linear sequence of states indicated by the reachability tree for an OCPN.

Liveness: Liveness implies the absence of deadlocks. A Petri net is live if, for all markings M in the reachability set of the initial marking $R(M_o)$, it is possible to fire any transition through some progressing firing sequence [20]. The OCPN models presented do not indicate any tokens and are therefore not live. To initiate an active OCPN process we can include an initial place with a single token in the OCPN model. By terminating the OCPN at the initial place we create a cyclic presentation rather than a strictly linear one. The result is a strongly connected graph, that is, one where there exists a directed path from every node to every other node in $P \cup T$. Since the reachability tree is linear and the OCPN construction process does not introduce any cycles, by introducing a single feedback loop (cycle) an OCPN is decidedly live. Additionally, liveness implies complete *coverability* for all M in $R(M_o)$.

Boundedness: A Petri net is k-*bounded* if the number of tokens at each place in a net does not exceed some k for any marking reachable from the initial marking M_o [20]. If the net is 1-bounded, it is called *safe*. Since OCPN's are built with a pairwise composition process without cycles, for every transition with two outgoing arcs there exists a corresponding transition with two incoming arcs, therefore, k-boundedness is assured. For every token created at a forking arc, one is absorbed at the joining arc, and the k tokens are preserved. Since $k = 1$ for OCPN's, they are safe.

Conservation: A Petri net is conservative if the number of tokens in the net does not decrease. For the OCPN, the number of tokens varies with the concurrency of the presentation processes, and therefore is not conservative. However, the initial and final number of tokens always remains equal to one, as discussed.

The objective for which we present Petri nets is the specification of multimedia process hierarchies indicated by temporal relationships. We introduce and prove theorems allowing the representation of process structures to present the synchronization of multiple streams of data. To provide a multimedia information service, the specification must be utilized in a manner to support the desired applicability. For an information storage and retrieval service, we must maintain the information necessary to synchronize pairwise any two data sources. This information clearly consists of the temporal relation T_R and the temporal intervals τ_α, τ_β, and τ_δ, between process entities, as we have demonstrated. Given this information, we can completely construct the synchronized process interaction, modeled by a Petri net. Recalling Section 2.2, between corresponding audio and image components the temporal relation *equals* is valid, with equal values for τ_α and τ_β. Between sequential image/audio pairs the relation *meets* holds with time values τ_α and τ_β corresponding to the durations of successive pairs of elements.

In Section 3, we describe how the OCPN model can lead to a storage schema for this information. In Section 4, an algorithm is presented for the extraction and utilization of the temporal information to generate a process structure indicated by the Petri net.

3 Database Architecture

In this section, we describe how an OCPN model is amenable to the construction of a logical database structure which can be utilized to allow synchronization and composition of multimedia entities. In Section 5, a detailed example of this process is described. Access methods are indicated to demonstrate how the synchronization schema allows general interfacing to multimedia information systems.

3.1 Database Schema

The modeling tool described in Section 2 allows us to build or describe complex process presentations. The OCPN implies temporal relationships between pairs of presentation objects (objects could be complex) and explicitly indicates resources and timing parameters for component objects. The Petri net is a convenient visual technique for capturing the specification of a multimedia presentation. A database schema is needed which preserves the semantics of the Petri net model to facilitate reproduction, communication, and storage of multimedia presentation.

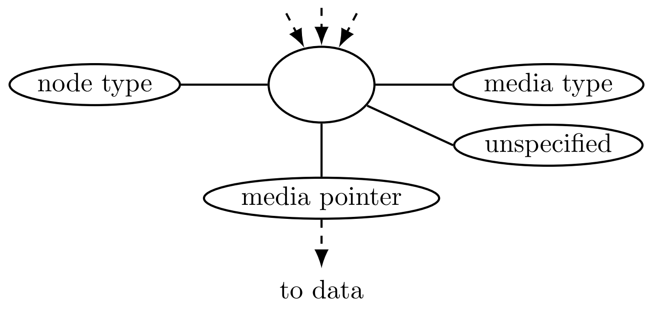
Because the OCPN construction technique is based on paring temporally related objects of arbitrary grain of synchronization, a database model which optimally handles a binary structure is desired. A hierarchical model is chosen for this purpose, therefore, schema development is influenced by the Petri net construction process and the tree-structured hierarchical model. We call the resultant schema a *synchronization schema*.

For multimedia data elements (not synchronization information) we can choose some more optimal database model based on the type of data, indexing scheme, or access paradigm. Optimality of multimedia database management is not addressed in this paper as we assume that the atomic multimedia data elements are readily accessible by links emanating from the synchronization schema. We do not assume any specific model for the multimedia database management component; rather, we build a framework which can be integrated, or used in a complementary fashion, with other DBMS techniques.

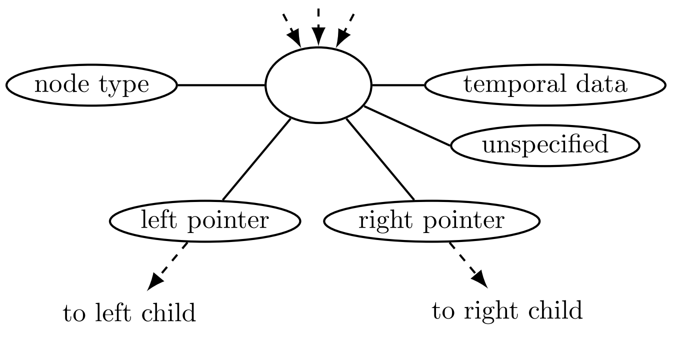
To capture the semantics of the OCPN model we need merely to pair, possibly complex, multimedia objects and identify them with temporal parameters τ_α, τ_β, and τ_δ. The OCPN, used to formally specify pairs of synchronized media elements is transformed into a database schema and database by selecting synchronized pairs of objects of the OCPN and assigning their temporal parameters to a database element. We design templates for the elements of the database schema to have the ability to capture this information using three types of tree nodes.

The first node type template is the leaf or *terminal* node as indicated in Figure 9a. This node has attributes which indicate node type (terminal, nonterminal, or meta type), media type (text, image, video, etc.), an additional unspecified attribute, and a pointer which indicates the location of the data for presentation. This node type indicates the elements at the finest grain for synchronization. Note that the in-degree of the terminal node has no restrictions. That is, the node can have many parents, facilitating links in time and space to multiple database schemata.

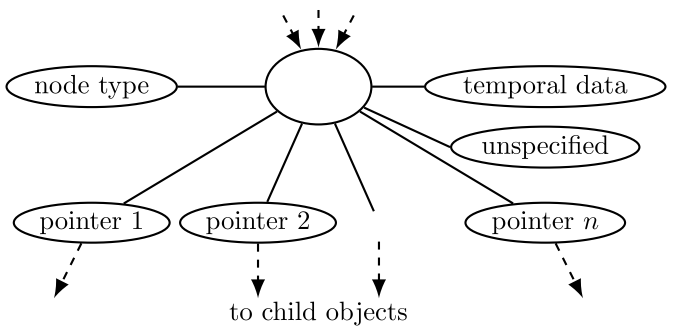
Nonterminal nodes have a slightly different structure as indicated in Figure 9b. Attributes for this template include node type, an unspecified attribute, left and right child pointers,



(a)



(b)



(c)

Figure 9: (a) *Terminal* node type template. (b) *Nonterminal* node type. (c) *Meta* node type template

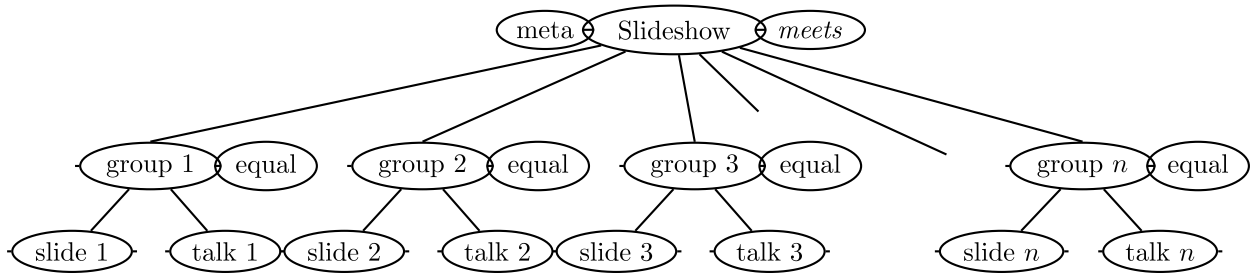


Figure 10: Slide presentation using meta node

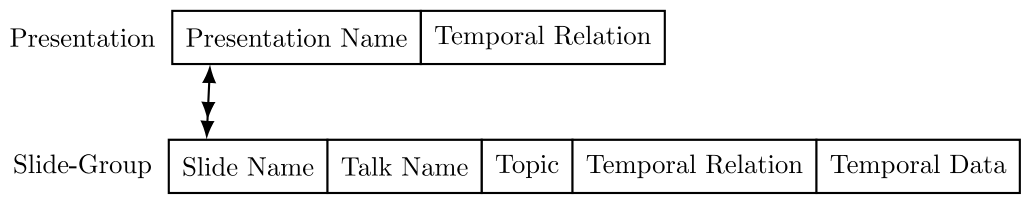


Figure 11: Slide presentation network schema

and temporal data (τ_α, τ_β, τ_δ, and the temporal relation T_R between the children). Again, in-degree can be arbitrary.

Two node types allow complete specification of the synchronization requirements, as indicated by the OCPN model. However, there are some applications (we will describe one in Section 3.2) in which it is not desirable to retrieve the entire object hierarchy upon access of the root element. Additionally, there can be a large amount of redundant information in the database with a binary tree representation and an equivalent relationship between all of the elements, indicating a benefit to describing a third structure for the description of a database schema.

Meta nodes, like the nonterminal nodes, have attributes node type, temporal data, and an unspecified attribute. Additionally, any number of pointers to other nodes are allowed. Like the other node types, in-degree is arbitrary. This node type is indicated in Figure 9c. Meta nodes associate many temporal intervals with a single temporal relation. They allow a compact representation of a common relation among many similar process intervals. In Figure 10 we indicate the association of many slide-talk pairs with a single meta node, sharing the temporal relation *meets*. In this example the number of nodes has been reduced from a potential $2n-1$ to $n+1$ with the introduction of a single temporal relation indicated by the meta node attribute *meets*.

Using the tree node types, pairs (or sets) of synchronized elements can be represented in synchronization schema developed for an OCPN. Consider Section 2.2 to clarify synchronization schema development. First, we identify and assign individual images and audio segments including time durations to nonterminal elements based on some useful attribute such as *topic*. Observing the OCPN of Figure 4, we can see points of synchronization corresponding to slide-talk pairs. Therefore, we select a parallel relation *equals*, with unique durations for each slide-talk pair, and form a set of synchronized pairs of objects using a nonterminal node template. In this case the assigned attributes are *Slide Name*, *Talk Name*, *Topic*, *Temporal Relation*, and *Temporal Data*. Since we have a set of nonterminal nodes related sequentially as indicated by the OCPN, we may group them using a meta node template, with *Presentation Name* and *Temporal Relation* as attributes. A hierarchical schema for this example is presented in Figure 11, where the meta node is identified as *Presentation* and the nonterminal nodes are identified by *Slide-Group*. The single-to-double headed arrow indicates a one-to-many relationship of a meta node in the hierarchical database model.

Based on the intended access/query methods, additional attributes can be assigned to the nodes of the database schema to support miscellaneous queries on subsets of the database hierarchy, such as a query on *Topic* of the schema of Section 2.2. For more complex OCPN representations, selection of parings of complex objects represents coarse-grain synchronization in the database. As objects become more refined at successive parings towards the terminal elements, synchronization becomes more fine-grained.

Database construction is the process of assigning actual data to the developed database schema. For a synchronization schema, we assume multimedia data elements are stored and managed by some entity which may be accessed by pointers from terminal nodes defined by the synchronization schema. Database construction begins by assigning multimedia data to the terminal nodes, including the required time durations of the data elements. To establish temporal parameters (time duration values) for nonterminal nodes or any complex objects, Lemma 1 is used by combining timing parameters of temporally related terminal elements and assigning them to their connecting nonterminal parent nodes. As this process is repeated, timing values are propagated up the database hierarchy until timing values are established at all nodes. With this completely established synchronization database, any complete connected subtree may be isolated and presented as a unit, with ignorance of relationships to other parts of the hierarchy. An interesting characteristic of this synchronization database is that any path from root to terminal element with observance of temporal relations and parameters can exactly determine the starting presentation time of the terminal element.

3.2 Synchronization Schema Access Methods

A multimedia information system can have many potential applications requiring multiple data access and retrieval methods tailored to each environment. We have proposed a schema construction process for the storage of objects based on presentation requirements in Section 3.1. In this section we indicate how several different access methods may be interfaced with a constructed synchronization schema.

By utilizing a hierarchical or network model for the representation of a synchronization schema we allow well established query techniques to be applicable to the schema. For example, since the synchronization schema can possess only one-to-one or one-to-many relationships characteristic of a network model, the schema fits into the CODASYL DBTG model [8]. By treating the attributes of the nodes as viable query parameters, we may perform conventional queries on subcomponents of the network schema without regard for the temporal relationships of the contained data. For example, using the CODASYL DBTG model we can perform the following query based on the synchronization schema (Figure 11) of Section 2.2.

```
presentation.name := “Friday’s Seminar”;
find any presentation using name;
find first slide-group within presentation-slide;
while DB-status = 0 do
    begin
        get slide-group;
        if slide-group.topic = “Petri Nets” then
            extract(slide-group.talk);
        find next slide-group within presentation-slide;
    end
```

This query retrieves all verbal annotations on the topic of “Petri Nets” presented in “Friday's Seminar." Since subtrees of the synchronization schema tree are temporally related, if a query of this kind were performed on a complex subtree rather than on a terminal element, retrieval would require synchronization of the data contained in the subtree. In Section 4, an algorithm is proposed to facilitate this synchronization from the database.

Multimedia information systems can be built with varying application and interface models. Of particular interest are ones based on hypermedia [30]. Ideally, data composition within these application environments can be simplified using the process synchronization model we propose. Hypermedia techniques can be directly adapted to a constructed synchronization schema and we demonstrate one such example in Section 5. A multimedia database will consist of many objects each consisting of diverse subobjects. Subobjects of a synchronization schema indicated by nonterminal nodes can be treated as independent objects, allowing comparison of attributes from many different sources. Using a separate hypermedia access entity, pointers to objects with particular attributes can be allowed, linking hypermedia references between objects and subobjects. Since subobjects can be presented independently of their object trees, reference to an object or subobject can result in subsequent retrieval and presentation of the stored data elements based on the maintained temporal parameters.

4 Object Retrieval and Presentation Algorithm

In the previous sections we indicated the tools and methods for specification of the time composition of multimedia objects, and for construction of a database system for their storage. In this section we describe the processing involved in the retrieval of stored multimedia entities as associated with our storage model and the multimedia information system of Figure 1.

Given a synchronization schema which includes temporal relations and timing values, we have two options for the timing data for which it contains:

1. Let our database contain only consistent markings with respect to timing. That is, allow timing values at the terminal nodes such that nonterminal and meta nodes posses timing values that are consistent with the temporal relations of the complete hierarchy. In Section 2.2, this would imply for a given image-annotation pair the durations of each element would necessarily be equal to satisfy the *equals* relation.

2. Allow nodes to have arbitrary timing values and relations; a more general case. Force consistency, if possible, at query or retrieval time by a processing algorithm, or otherwise abuse the timing relationships between objects and presentation elements.

We choose option 1) and restrict the combination of objects to ones that can be logically related. All objects can be related based on at least one temporal relation; however, as an example, it is not possible to give two temporal intervals with different durations a temporal relationship of *equals*. A consideration of the coercion of temporal relationships for an inconsistent marking will be presented in a future work.

Assuming a consistent database, we present an algorithm which presents multimedia data elements based upon the original Petri net model of the multimedia presentation. The algorithm builds a process tree which indicates the Petri net model used to specify the multimedia presentation. The algorithm is recursive and can be applied to a schema of arbitrary complexity. The basic algorithm is as follows.

Presentation Algorithm:

```
Step 1) For object Object, identify attributes (children, node type, etc.)

Step 2) If Object is a nonterminal then evaluate its temporal relations:

  Step 2.1) If temporal relation is sequential (before or meets) then:
      Step 2.1.1) Recursively invoke presentation algorithm on left-child object.
      Step 2.1.2) Delay $\tau_\delta$.
      Step 2.1.3) Recursively invoke presentation algorithm on right-child object.
  Step 2.2) If temporal relation is parallel (overlaps, $during^{-1}$, starts, $finishes^{-1}$, or equals)
            then:
      Step 2.2.1) Create a new process thread and recursively invoke presentation algo-
                  rithm on left-child object.
      Step 2.2.2) Delay $\tau_\delta$.
      Step 2.2.3) Create a new process thread and recursively invoke presentation algo-
                  rithm on right-child object.

Step 3) If Object is a terminal then present data on the appropriate I/O device for duration
        indicated by timing parameter.

Step 4) Terminate when all recursive invocations have completed.
```

Since the synchronization database contains temporal relations, we can identify the nature of the processing concurrency. All of the relations fall into one of two categories: sequential or parallel. With the presentation algorithm we build a process tree which possesses the same connectivity as the database schema and therefore the same process execution as the underlying Petri net. The algorithm parses the schema tree in a top-down, left-right, recursive manner beginning from any node and generates a process tree for object presentation. As each node is parsed, connectivity of the process tree is maintained identical to the database schema tree with concurrency necessary to perform process execution as specified by the OCPN. To cover all temporal relations in a uniform manner with a left-right pass we identify the process with initial execution prior to the potential delay process from the OCPN models of the temporal relations. The first process is assigned to be the left child, and the later process (if any) is assigned to the right child. To maintain consistency with our earlier notation we require the use of the inverse temporal relations for *during* and *finishes* to place the earlier processes on the right side of the tree for our algorithm. As per the Petri net model, we let the right child represent process P_α, and the left child represent process P_β.

Observe the grain of synchronization achieved between pairs of processes. Pairs of processes are synchronized within an invocation of the algorithm. Since the parent process (presentation algorithm) cannot terminate until all of its children have terminated, the wait

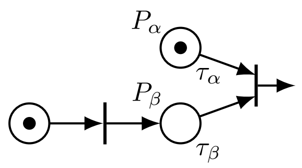


Figure 12: Process waiting

and subsequent termination of a call to the algorithm represents the waiting and firing of a transition in the Petri net model in which two arcs join. See Figure 12.

To this point, we have considered terminal and nonterminal nodes. Meta nodes allow groups of related temporal hierarchies to be presented with a common temporal attribute. This node type facilitates repetitive retrieval of temporal structures. Out-degree of this type of node can be greater than two, permitting one-to-many relationships necessary for a network database model. Components of a meta node are all related by the same temporal attribute, for example *meets* or *equals*. We permit meta nodes to prevent the construction of very large binary trees, as discussed in Section 3.1. The repetition of the sequential or parallel processes indicated by the meta node is facilitated by a trivial modification to the presentation algorithm. This change allows spawning or sequencing of the necessary child processes, which is not presented, for lack of space.

The algorithm presented facilitates the retrieval and subsequent presentation of stored multimedia entities. Specifically, the algorithm builds a process network which replicates the original Petri net model used to specify the multimedia presentation, assuming a consistent database with respect to timing. In Section 5, we develop a database schema for which the algorithm is applied.

5 Application Example

In this section, we present a detailed example which elaborates the ideas and models presented in the preceding sections. The example contains both static and dynamic data in the form of audio, image and textual data which require synchronization at the presentation level.

Example 2:Anatomy and Physiology Instructor

The anatomy and physiology instructor is a simple multimedia application example based on the hypermedia paradigm and temporal relation specification. Figure 13 indicates the coarse grain hypermedia network for the anatomy and physiology instructor. We assume that arcs in this figure represent links between informational units, represented by nodes, which contain lessons. Beginning at the **Start** node, users can browse via link selection or formulate explicit queries against the overall database.

When browsed or selected, information at a node is activated, causing its subsequent retrieval and presentation. For example, browsing **Alveolar Ventilation** assuming an initial starting point of **Respiration**, causes the information associated with that node to be presented. Fine grain synchronization is performed within the informational units.

A database structure is developed at two levels. At the coarse grain we desire the ability to

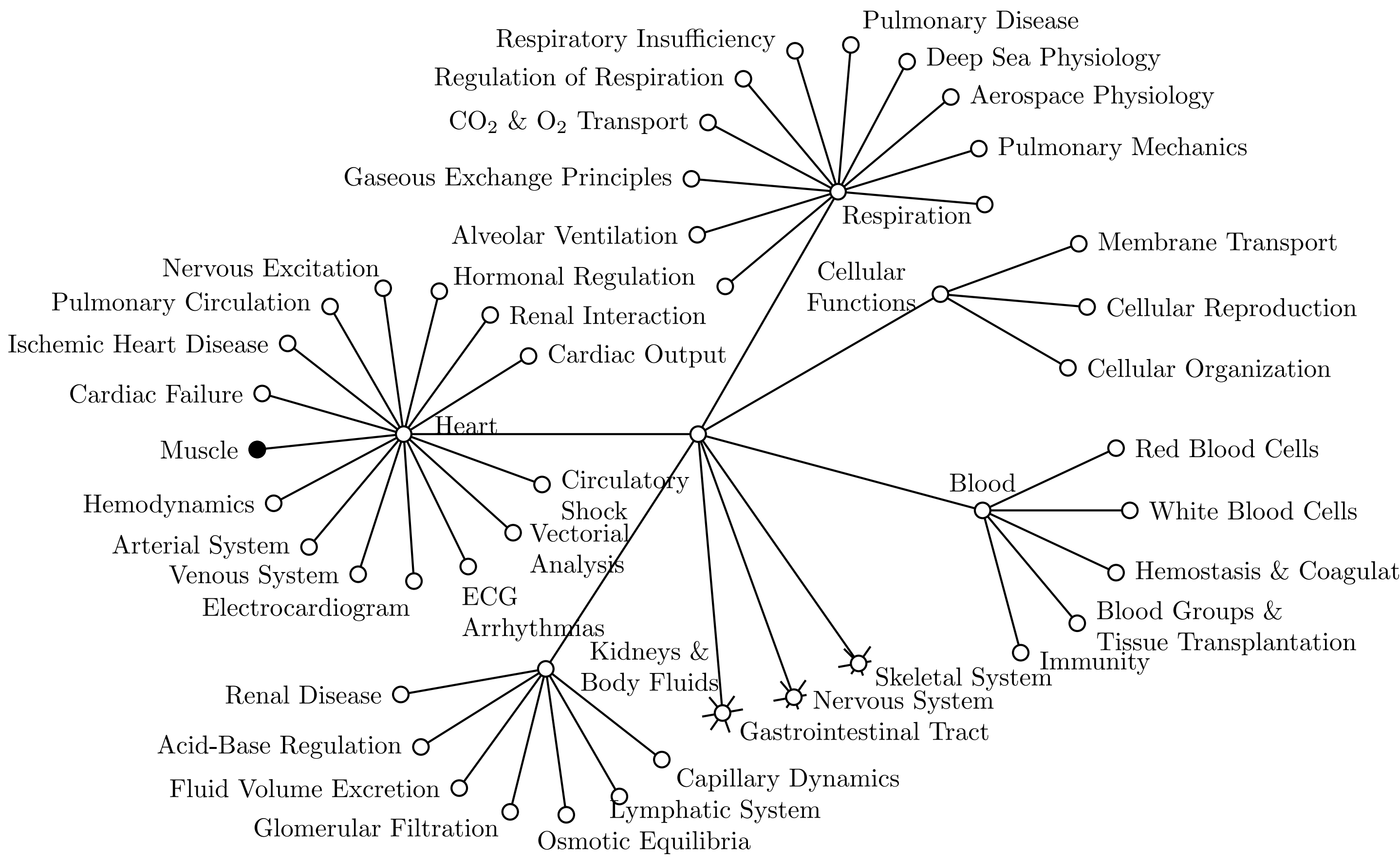


Figure 13: Anatomy and physiology instructor information network [11]

select informational units such as **Heart** or **Kidneys**. At the fine grain the synchronization constraints must be specified for concurrent presentation of various media elements. Each informational unit can have different presentation requirements requiring distinct synchronization specification. Ideally, one specification suffices for all informational units, simplifying the task of presentation design. For the general case, all nodes have different characteristics. We describe the synchronization requirements for one informational unit: **Heart Muscle**.

To specify the synchronization requirements within an informational unit a model for the presentation of information is required. For the example, the model used is based on the sample workstation screen indicated in Figure 14.

In this figure the informational unit **Heart Muscle** is presented. Note the different regions associated with the various media types including audio, video, text, image, and animated image (Views). Let us assume some characteristics and relationships between elements for this example. Figure 15 indicates possible relationships in time of the various elements of the example. This timeline representation shows that the textual component, the icon, and the "location in body" images should persist for the duration of the informational unit presentation. The views are specified to change with time to provide a gradual rotation, or animation, of the organ selected, while the video and associated audio components begin presentation after a constant delay of τ_{13}. On the spatial axis are indicated the resources r_1

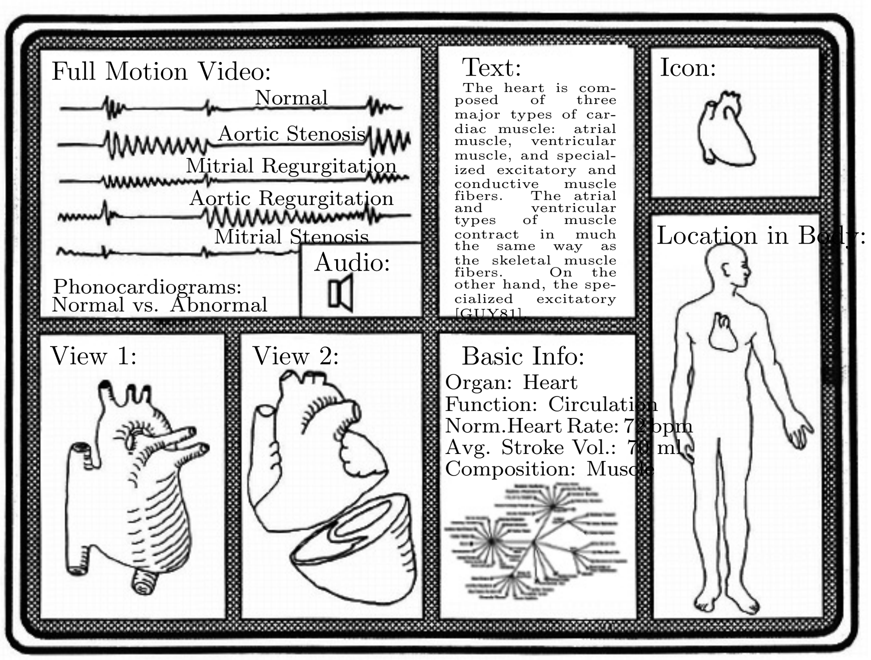

Full Motion Video:
Normal
Aortic Stenosis
Mitrial Regurgitation
Aortic Regurgitation
Mitrial Stenosis
Phonocardiograms:
Normal vs. Abnormal
Audio:
Text:
The heart is composed of three major types of cardiac muscle: atrial muscle, ventricular muscle, and specialized excitatory and conductive muscle fibers. The atrial and ventricular types of muscle contract in much the same way as the skeletal muscle fibers. On the other hand, the specialized excitatory [GUY81]
Icon:
Location in Body:
View 1:
View 2:
Basic Info:
Organ: Heart
Function: Circulation
Norm.Heart Rate: 72 bpm
Avg. Stroke Vol.: 70 ml
Composition: Muscle

Figure 14: Anatomy and physiology instructor workstation instance

through r_8, associated with the workstation screen regions and the audio output.

Synchronization of the media elements is specified by the OCPN model. By combining elements of the presentation in temporally related pairs, we create a OCPN for the information unit shown in Figure 16. This OCPN indicates the processing required for the presentation of a single anatomy and physiology instructor information unit.

Between transition t_1 and t_9, we see a maximum of eight threads of presentation corresponding to the timeline of Figure 15. At the outset, seven threads are created as indicated by the initial transition, t_1. Places p_{16} through p_{19} represent static data which are active for the duration of the presentation. The two interacting threads, p_1, p_3, p_5, p_7, p_9, p_{11}, and p_2, p_4, p_6, p_8, p_{10}, p_{12}, represent the animation of two views of the organ, synchronized at points indicated by t_4, t_5, t_6, t_7, t_8, and t_9 of the OCPN. The places p_{14} and p_{15} represent dynamic audio and video data, delayed by τ_{13}.

Having specified the synchronization requirements of an informational unit, we create a network database schema to store the medical data and temporal relationships between data items. This structure is built directly from the pairs chosen in OCPN development. Choosing $View_1$ and $View_2$ pairs based on the temporal relation *equals* forms a nonterminal node of the database schema, which we call *organ-horizontal-vertical-view*. Text elements, still images, and video/audio are similarly combined using the *equals* relation generating nonterminal nodes *organ-text-info*, *organ-icon-location*, and *organ-video-image*, respectively. These nonterminal nodes are related by further pairing and temporal relation assignment. *Organ-horizontal-vertical-view* is appended to a meta node; *organ-views*, with temporal attribute *meets*, allowing representation of multiple image pairs. Relating the nonterminal nodes results in additional nonterminal nodes. *Organ-image* relates *organ-views* and *organ-video-image* with the *finishes* relation, accounting for time delay τ_{13}. Similarly, *organ-text-icon* is

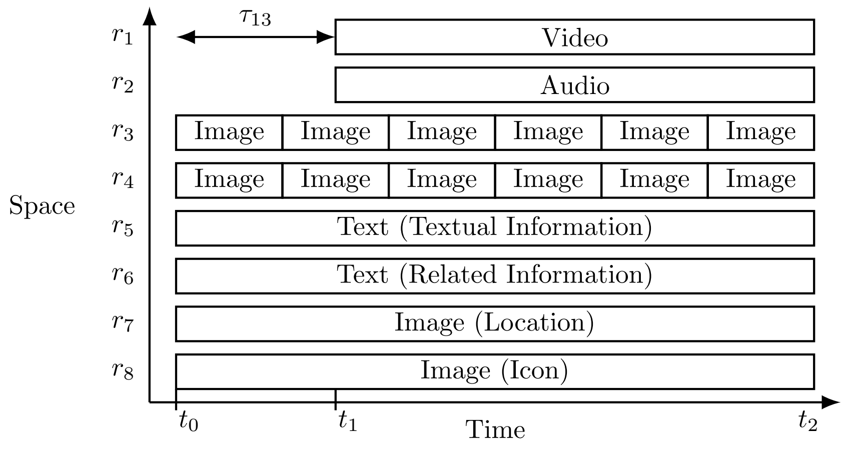


Figure 15: Anatomy and physiology instructor timeline

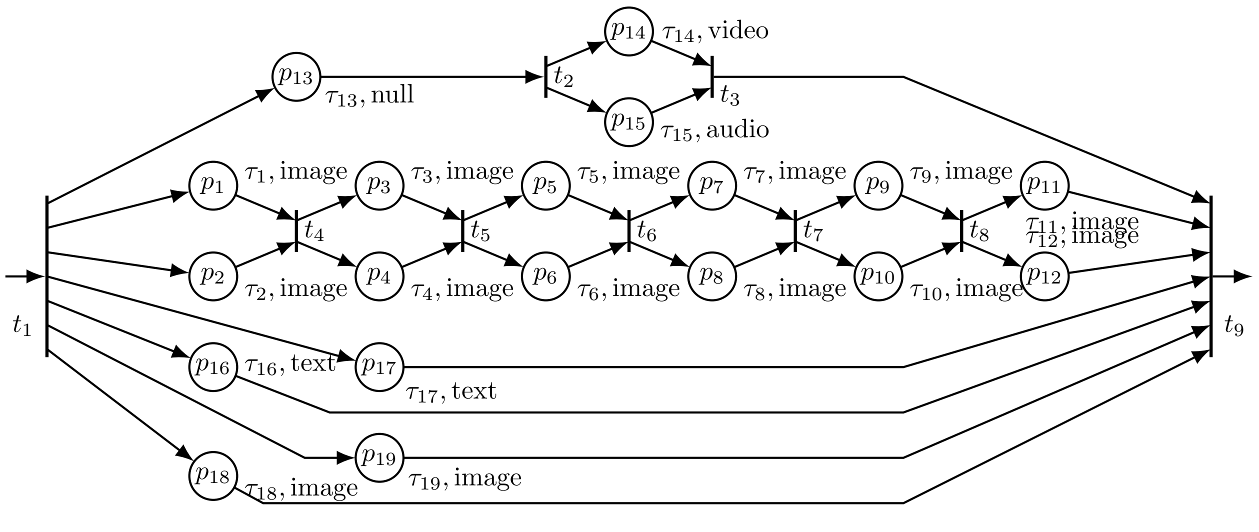


Figure 16: Anatomy and physiology instructor OCPN

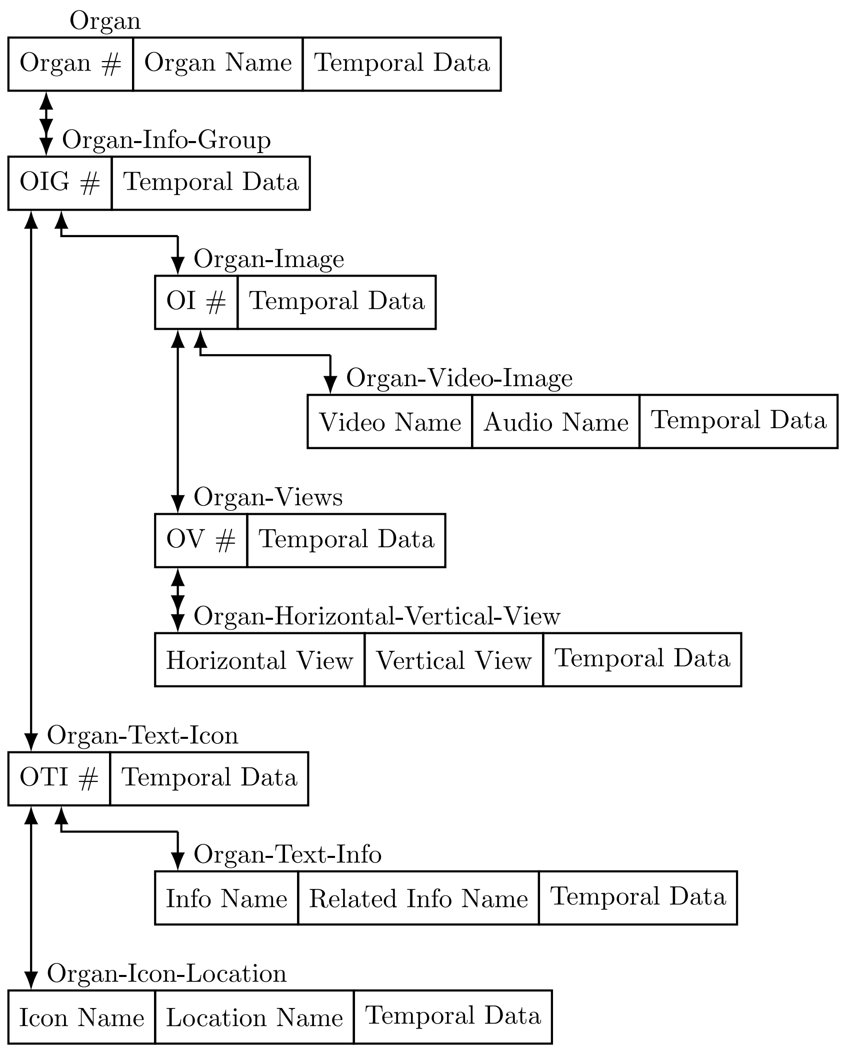


Figure 17: Anatomy and physiology instructor network schema

related to *organ-image* by *organ-info-group*. Finally, *Organ*, a meta node, represents the set of organs with a similar schema for which **Heart Muscle** is a member. Terminal nodes, not specified here, indicate the exact location of stored data instances within the medical database. The resultant schema, presented in Figure 17, is composed with a meta node at its root to facilitate selection via query or hypermedia browsing. In this case we assume that all organs have a similar temporal synchronization requirement.

In assigning data to the database schema we must maintain temporal consistency. According to our OCPN model, the pairings we have chosen, and the temporal relations used, we have a set of consistency requirements:

$$\begin{aligned}\text{Duration of unit} &= \tau_1 + \tau_3 + \tau_5 + \tau_7 + \tau_9 + \tau_{11} \\ &= \tau_{13} + \tau_{14} \\ &= \tau_{16} = \tau_{17} = \tau_{18} = \tau_{19} \\ \tau_1 &= \tau_2\end{aligned}$$

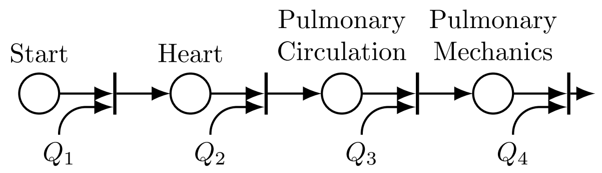


Figure 18: Browsing the anatomy and physiology instructor

$$\tau_3 = \tau_4$$
$$\tau_5 = \tau_6$$
$$\tau_7 = \tau_8$$
$$\tau_9 = \tau_{10}$$
$$\tau_{11} = \tau_{12}$$
$$\tau_{14} = \tau_{15}$$

By storing data which satisfy these constraints, the system performs as specified by the OCPN, without delays caused by inappropriate values. Note that system delays associated with communication or implementation are ignored and must be incorporated into the model.

Both *organ* and *organ-view* are represented as meta nodes with one-to-many relationships. The screen of the workstation, as indicated in Figure 14, represents an instance of the database described by the network schema.

Consider the interaction of a user with this multimedia system. If we use subnet replacement, we can describe any node in the network graph as an equivalent Petri net place. Browsing or selecting nodes permits the sequential access of informational units. By representing the selection activity as a query, a Petri net description of browsing a particular path is formulated. Figure 18 shows a net indicating the sequential access of the information units **Heart**, **Pulmonary Circulation**, and **Pulmonary Mechanics** in response to queries Q_1, Q_2, and Q_3. Using this notation one could assign bidirectional Petri net transition at each link (arc) in the information network to describe all possible browsing actions. A complete Petri net description of the application results.

We next explain the application of the presentation algorithm to the example information unit **Heart Muscle**. We demonstrate the algorithm as a sequence of steps as follows:

1. Query **Heart Muscle**, resulting in selection of the appropriate *organ-info-group*.

2. Call the presentation algorithm initially for object parameter **Heart Muscle**.

3. Fetch and identify attributes, indicating nonterminal subobjects and that recursive calls are required.

4. Since concurrency is indicated (*organ-text-icon equals organ-image*), create two processes threads with no delay between them ($\tau_\delta = 0$), each calling the presentation algorithm recursively for the right and left subobjects.

5. Invocations of the presentation algorithm for *organ-text-icon* and *organ-image* are further split into subprocesses until the terminal nodes are attained.

6. Terminal nodes point to actual multimedia data elements. Data is fetched from databases and presented at the workstation based on the duration indicated by the stored timing parameter.

7. Presentation processes complete and terminate invocations of the algorithm. The initial invocation of the algorithm terminates as the children terminate, as per transition t_9 in OCPN model.

The examples in this section clearly illustrate how we can use the OCPN modeling strategy for many scenarios. However, there are some limitations to our current strategy which must be extended.

6 Concluding Remarks

Several issues are not addressed by our model. First, we ignore spatial considerations required for composition of multimedia objects. For example, our model does not specify where to put multiple images on the workstation surface. One possible solution is to assign spatial characteristics to the resource component of the OCPN model, thereby fully defining multimedia data integration with the OCPN. However, treatment of the occurrence of spatial clashes still remains (two processes require same resource: window on the screen).

Second, user input to a multimedia service is not well described by the OCPN model. For instance, stoppage of a dynamic presentation is not modeled. We also ignore the possibility of a reverse presentation of the multimedia objects. Certainly we can model the reverse presentation of an object, but our current algorithm does not have a provision to support this activity. Similar questions arise when we examine the possibility of manipulations of object elements during display, for example, zooming on an image currently under presentation. This kind of operation also does not fit well into our model or processing algorithm.

Considering editing and updating multimedia objects which are of the dynamic type we must realize that objects are constructed based on an original OCPN model of the presentation requirements which have been translated to a database schema. If radical changes are desired, it is necessary to modify the database schema. A more suitable approach is to use an object-oriented approach that would allow dynamic redefinition of the database schema. Editing, including combination of multimedia objects with diverse presentation timing requirements is likely to introduce temporal inconsistencies, as discussed in Section 4. What provisions are necessary to preserve a coherent presentation? Certainly some elements can be delayed to preserve others, for example, still images to preserve voice.

In Section 4, we chose to restrict the combination of objects to ones that maintain consistency of temporal relations. The proposed schema allows for groups of carefully designed entities to be presented together which have been choreographed in advance, using the OCPN. If mixed, nonplanned objects are combined, timing can be distorted. However, since most relations will be *meets*, *equals*, or *starts* relations, little inconsistency will be created. We are addressing the issue of time coercion of temporal relations in a future work.

There are a number of issues which must wait for a subsequent work. We presented an approach to storage of synchronization information for complementary use in association, or integrated into a data management facility for actual multimedia elements. The proposed synchronization schema construction needs to be examined from the viewpoint of functional, object, and access-oriented paradigms for rigorous integration with multimedia database management capabilities. These methods may reveal a superior approach to interfacing general multimedia services to a database.

We describe a multimedia data composition problem in this paper. related to this problem is the decomposition of multimedia objects and their distribution across a decentralized system. What impact the distribution has on the retrieval and composition of objects is yet to be investigated. Of particular interest are timing delay considerations introduced by implementation.

Recapitulating, we propose the tools and strategy for the formal specification of multimedia object composition with respect to synchronization. Using a form of timed Petri net, called a OCPN, multimedia objects can be choreographed in time and stored in a database indicated by a schema developed using the OCPN. This database is utilized as a storage element for synchronization supporting multiple access techniques including conventional query systems and hypermedia. We present examples to support our models, indicating the usefulness in building multimedia information services.

Acknowledgement

This research was supported in part by the NYNEX Corporation through the New York State Center for Advanced Technology in Computer Applications and Software Engineering (CASE) at Syracuse University.

References

[1] Agerwala, T., “Putting Petri nets to Work,” *IEEE Computer*, December 1979, pp. 85–94.

[2] Allen, J. F., “Maintaining Knowledge about Temporal Intervals,” *Communications of the ACM*, November 1983, Vol. 26, No. 11, pp. 832–843.

[3] Anderson, T. L., “Modeling Time at the Conceptual Level,” *Improving Database Usability and Responsiveness*, P. Scheuermann, Ed., Academic Press, New York, 1982, pp. 273–297.

[4] Bolour, A., Anderson, T. L., Dekeyser, L. J., Wong, H. K. T., “The Role of Time in Information Processing,” *ACM SIGMOD Record*, Vol. 12, No. 3, 1982, pp. 27–50.

[5] Bubenko, J. A., Jr., “The Temporal Dimension in Information Modeling,” *Architecture and Models in Data Base Management Systems*, G. M. Nijssen, Ed., North-Holland, New York, 1977, pp. 93–118.

[6] Commoner, F., Holt, A. W., Even, S., Pnueli, A., “Marked Directed Graphs,” *Journal of Computer and Systems Science*, Vol. 5, October 1971, pp. 511–523.

[7] Coolahan, J. E., Jr., Roussopoulos, N., "Timing Requirements for Time-Driven Systems Using Augmented Petri Nets," *IEEE Trans. on Software Engineering*, Vol. SE-9, No. 5, September 1983, pp. 603–616.

[8] "CODASYL Data Base Task Group April 71 Report," *ACM*, New York, NY, 1971.

[9] Conklin, J., "Hypertext: An Introduction and Survey," *IEEE Computer*, September 1987, pp. 17–41.

[10] Garcia-Luna-Aceves, J. J., Poggio, A., "Multimedia Message Content Protocols for Computer Mail," *Proc. IFIP WG 6.5 Working Conf. on Computer-Based Message Services*, Nottingham England, May 1984, Hugh Smith, Ed., North-Holland, New York, pp. 87–98.

[11] Guyton, A. C., *Textbook of Medical Physiology*, W. B. Saunders Co., Philadelbphia, 1981.

[12] Hamblin, C. L., "Instants and Intervals," *Proc. of the 1st Conf. of the Intl. Society for the Study of Time*, J. T. Fraser, et al., Ed, Springer-Verlag, New York, 1972, pp. 324–331.

[13] Hoare, C. A. R., "Communicating Sequential Processes," *Communications of the ACM*, Vol. 21, No. 8, August 1978, pp. 666–677.

[14] Hodges, M. E., Sasnett, R. M., Ackerman, M. S., "A Construction Set for Multimedia Applications," *IEEE Software*, January 1989, pp. 37–43.

[15] Holliday, M. A., Vernon, M. K., "A Generalized Timed Petri Net Model for Performance Analysis," *Intl. Conf. on Timed Petri Nets*, Torino, Italy, July 1985, pp. 181–190.

[16] Ichbiah, J., et al., "Reference Manual for the Ada Programming Language," Technical Report, United States Department of Defense, July 1980.

[17] International Organization for Standardization, ISO Document No. 8613, ISO, Geneva, March 1988.

[18] Kahn, K., Gorry, G. A., "Mechanizing Temporal Knowledge," *Artificial Intelligence*, Vol. 9, No. 1, August 1977, pp. 87–108.

[19] Lippman, A., "Movie-Maps: An Application of the Optical Videodisc to Computer Graphics," *Computer Graphics*, July 1980, (*SIGGRAPH '80 Conf. Proc.*, Seattle, WA, July 1980, pp. 32–42).

[20] Peterson, J. L., "Petri Nets," *Computing Surveys*, Vol. 9, No. 3, September 1977, pp. 225–252.

[21] Poggio, A., Garcia Luna Aceves, J. J., Craighill, E. J., Moran, D., Aguilar, L., Worthington, D., Hight, J., "CCWS: A Computer-Based Multimedia Information System," *IEEE Computer*, October 1985, pp. 92–103.

[22] Postel, J., “A Structured Format for Multimedia Documents,” Report RFC767, DDN Network Information Center, SRI International, Menlo Park CA, 94025, USA, August 1980.

[23] Razouk, R. R., Phelps, C. V., “Performance Analysis using Timed Petri Nets,” *Proc. 1984 Intl. Conf. on Parallel Processing*, August 1984, pp. 126–129.

[24] Segev, A., Shoshani, A., “Logical Modeling of Temporal Data,” *Proc. ACM SIGMOD Conf. on Management of Data*, San Francisco, May 1987, pp. 454–466.

[25] Snodgrass, R., Ahn, I., “A Taxonomy of Time in Databases,” *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, Chicago, June 1988, pp. 236–246.

[26] Snodgrass, R., “The Temporal Query Language TQuel,” *ACM Trans. on Database Systems*, Vol. 12, No. 2, June 1987, pp. 247–298.

[27] Studer, R., “Modeling Time Aspects of Information Systems,” *Intl. Conf. on Data Engineering*, Los Angeles CA, February 1986, pp. 364–373.

[28] Sventek, J. S., “An Architecture for Supporting Multi-Media Integration,” *Proc. IEEE Computer Society Office Automation Symposium*, April, 1987 pp. 46–56.

[29] Tansel, A. U., Arkun, M. E., Ozsoyoglu, G., “Time-by-Example Query Language for Historical Databases,” *IEEE Trans. on Software Engineering*, Vol. 15, No. 4, April 1989, pp. 464–478.

[30] Yankelovich, N., Meyrowitz, N., van Dam, A., “Reading and Writing the Electronic Book,” *IEEE Computer*, October 1985, pp. 15–30.

[31] Zuberek, W. M., “Performance Evaluation Using Extended Timed Petri Nets,” *Intl. Conf. on Timed Petri Nets*, Torino, Italy, July 1985, pp. 272–278.