AUTHORING SYSTEMS

Advances in technology allow for the capture and manipulation of multiple heterogeneous media, and so the demand for these media in digital multimedia documents is a natural progression. Multimedia documents containing heterogeneous media require complex storage, editing, and authoring tools. While there are numerous document standards, few have addressed all of the requirements of documents containing multiple, time-based media. Furthermore, representing document structures and playback scenarios visually is challenging, and only a small number of effective tools have emerged in both the commercial and research realms. Authoring systems for these documents must aid in both the spatial and temporal layout of media and may also have to support the modeling of user interactions with time-based media. This article discusses multimedia document standards, visual techniques for document authoring, and research issues for the future of authoring systems, including the impact of networked mobile agents.

MULTIMEDIA DOCUMENTS AND TERMINOLOGY

Structuring and laying out information for human consumption can be a difficult task. Publishers of newspapers and magazines face this challenge regularly as they lay out text and graphics onto printed pages. Publishers of multimedia content for human consumption via computer screens face even greater challenges because of both *time* and *interactivity*. Unlike a magazine and more like a movie, a multimedia presentation may have timing constraints. Furthermore, unlike both magazines and movies, multimedia presentations can be interactive and so their perceived presentation may change each time they are experienced.

Before the technological advances of digital computers, a *document* usually meant a *book*. Authoring such a document required writing a linear story line that described events through time—the book provides the exact same story each time that it is read. Creating a multimedia presentation is considerably more complex. Unlike a book, which we may call monomedia and linear, multimedia presentation may use media that must occur simultaneously or in some related way; all these relations must be specified by the author. Thus from the authoring point of view, there is a need for methods and models of representing temporal relations in multimedia documents.

One of the main issues in temporal models of documents is the model's flexibility to express different temporal relationships. Throughout this article, we are mainly concerned with the temporal behavior of documents, and other document attributes—including its layout, quality, and playback speed—are not the primary focus of our investigation. Of main issue is the representation and modeling of multimedia scenarios that "play themselves back" as well as letting the user interact with the running presentation, thereby driving it in a custom direction. Our focus is on those tools allowing authors to create scenarios that offer these nonhalting, transparent options to viewers.

To begin to understand the problem, consider Table 1. Various media types are categorized as having a temporal nature or not. If a media element has intrinsic attributes that relate directly to time-related qualities, it is considered a temporal media element. For example, since a video has a frame rate and an audio has a sample rate, they are both considered temporal. Text and images have no such attributes. However, as the third row of the table suggests, in a multimedia context any media type can be assigned temporal attributes in the form of *relationships*. An image, for example, can be programmed to be rendered only between the times 10 s and 15 s. Thus, the image has a *duration* of 5 s. Furthermore, the temporal attributes of a media element may be related to other media as well as to time. For example, a text (e.g., a title) may be programmed to appear on the screen exactly 10 s after the appearance of a logo graphic. Finally, the temporal

Table 1. Various Media Types and Their Temporal Classification

Media Classification	Text	Image	Graphic	Video	Audio	Animation
Static media	Yes	Yes	Yes	No	No	No
Temporal media	No	No	No	Yes	Yes	Yes
Temporal in a multimedia document context	Yes	Yes	Yes	Yes	Yes	Yes

attributes of a media may be related to events that occur in a totally asynchronous fashion. For example, a logo may be programmed to appear anytime that the user moves the mouse over the top of a small icon. The act of moving the mouse over the icon is called an *event* and it triggers the rendering of the logo graphic. This type of event is *asynchronous* because not only does the author of the multimedia presentation not know when it will happen, but the author does not even know if it will happen at all. Table 2 summarizes the type of relationships that can occur between media in a multimedia document (see also row 3 in Table 1).

Finally, we introduce the concept of the multimedia document and a sample of how one might look graphically. Figure 1 illustrates a multimedia document. When this document about Africa starts, the title, a video, and textual subtitles in synch with the video all start to play. At a particular point in the presentation, say between 10 s and 20 s, the video mentions African wildlife. At any time during this 10-s "window," the user is able to make a mouse selection on the "?" icon and switch to a short video that relates to African wildlife. This interaction is asynchronous since its time is unknown in advance. If the choice is not made the original multimedia presentation continues "on its track."

In this article, we make use of terminology, some of which originate elsewhere (1,2). Some required definitions are as follows:

- **Events:** Points at which the display of media objects (text, video, etc.) can be synchronized with other media objects. We focus on *start* and *end* events but we can generalize to *internal* events.
- **Synchronous events:** Those with predictable times of occurrence (e.g., their temporal placement is known in advance).
- **Asynchronous events:** Those with unpredictable times of occurrence and duration (e.g., their time of occurrence cannot be known in advance).
- **Temporal equality:** A synchronization constraint requiring that two events either occur simultaneously or that one precedes the other by a fixed amount of time.
- **Temporal inequality:** A synchronization constraint requiring, for example, that for two events, A and B, they occur such that A precedes B by an unspecified duration, by at least some fixed time, or by at least some fixed time and at most another fixed time.
- **Hypermedia:** Implies store-and-forward techniques where user actions such as mouse selections on hot spots cause the system to retrieve a new "page" of data, which could be an image, text, video, etc. There are usually no temporal relationships between media.
- **Passive multimedia:** Implies a fully synchronized document that "plays itself back" through time, synchronizing all media objects together.
- **Active multimedia:** Implies that there are hypermedia-type choices presented to users during the playback of a multimedia document that allow the user's interaction to "drive" the playback.
- **Scripting language:** A language such as SGML (Standard Generalized Markup Language) or one of its instances, such as HTML (Hypertext Makeup Language), that allows the addition of semantic and logical information to data using a markup language.
- **Scenario:** A term used for describing a completely specified (e.g., authored) multimedia presentation.

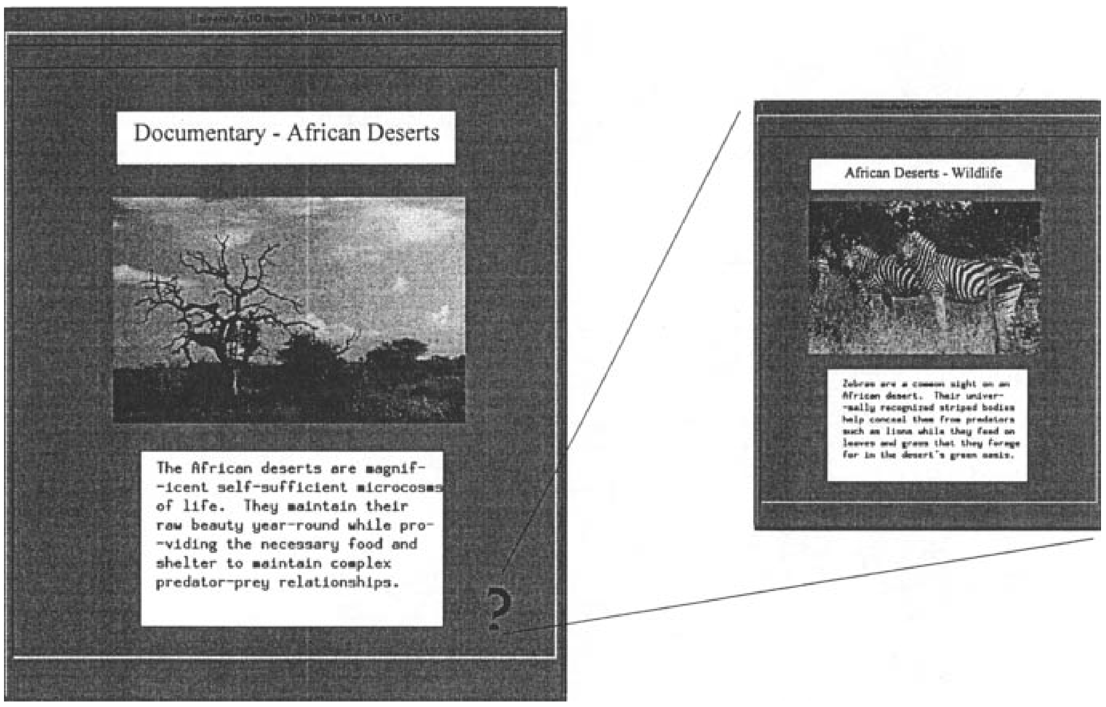
Table 2. Different Types of Temporal Relationships That May Exist Between Media in a Multimedia Document

Relationship Type	Description
Synchronous	The media occurs at a specific time (relative to the starting reference time of 0 s) for a specific duration. For example, image I2 occurs at time 20 s for 10 s.
Relative	The media occurs relative to a temporal attribute of another. For example, image I2 occurs 5 s after the rendering of graphic G3.
Asynchronous	The media occurs relative to an event whose occurrance is not even guaranteed or, if it is, the absolute time of the event is not known in advance. For example, the image I3 occurs if and when the mouse is moved over an icon I4.

It should now be clear that both multimedia documents and the process of authoring them are complex. With inter- and intramedia relationships and asynchronous events, the onus is clearly on both the logical structure of the document and the authoring tool to aid the author in creating such a complex renderable entity. The remainder of this article covers the following topics: (1) existing and de facto standards related to multimedia documents, (2) visual techniques for document authoring, (3) recent research achievements, and (4) the impact of the mobile agent paradigm on authoring systems.

MULTIMEDIA DOCUMENT STANDARDS

This section provides a brief overview of several key standards for document architecture, multimedia data format, and markup language standards for presentational applications. Standardization of these issues directly affects multimedia authoring systems. For example, authoring systems may be compliant to one particular standard (e.g., produce



Documentary - African Deserts

The African deserts are magnif-
-icent self-sufficient microcosms
of life. They maintain their
raw beauty year-round while pro-
-viding the necessary food and
shelter to maintain complex
predator-prey relationships.

?

African Deserts - Wildlife

Zebras are a common sight on an
African desert. Their univer-
-sally recognized striped bodies
help conceal them from predators
such as lions while they feed on
leaves and grass that they forage
for in the desert's green oasis.

Figure 1. A sample active multimedia document. A title box, video, and accompanying subtitles are presented. The appearance of the question mark denotes a "temporal window" in which the user may or may not make a selection to change the course of the presentation.

documents that conform to ODA [Open Document Architecture] or HTML) or may be compliant to none. Furthermore, since standards affect logical document structure they may also affect the graphical user interfaces used to author those documents and are therefore in context with this article.

Both SGML (3) and ODA (4) were designed to facilitate the representation and exchange of documents. They were tailored for subtly different settings—ODA for the office and SGML for a publishing environment. SGML represents a document as a grouping of logical elements. The elements are composed hierarchically and together they form the logical structure of the document. SGML uses tags to mark up the text and create elements. Markup is distinguished from regular text by enclosing it in braces (i.e., ⟨Chapter⟩) and the end of a logical element is marked up with a slash as well as the braces (i.e., ⟨/Chapter⟩). Furthermore, logical elements may have attributes—title, chapter, etc. A document created with SGML refers to some document class stored as a Document Type Definition (DTD). It determines the structure of the document, but users may define their own DTDs. Furthermore, DTDs are developed in an object-oriented fashion so prototyping is facilitated by code re-use and DTD class specializations.

ODA is a more robust type of architecture. ODA provides tools for specifying how the document should be laid out on the page, whereas SGML does not. ODA documents consists of a profile and content. The profile contains attributes of the document (e.g., title, chapter), and the content of the document is the combination of text and graphics specified by the author, as well as two important structures, the logical and layout structures. At the top of the hierarchic logical structure is the logical root and at the bottom are the basic logical objects that are atomic. The layout structure is also hierarchic, consisting of page sets, frames, and blocks that are laid out on the printed page as desired by the author. Layout structures are object oriented so that more than one may be defined for a given document. Specific layout structures are the instances of generic layout structures that are the templates for documents. An author can specify layout styles and so, for example, it is possible to specify that each figure should start on a new page. Neither SGML nor ODA can handle temporal information. HTML (5) is an SGML DTD created for the purpose of enabling a worldwide distributed hypertext system. HTML has become a de facto standard for on-line Internet documents and is the native language of the World Wide Web (WWW) and its browsers, such as Netscape's *Navigator* and Microsoft's *Internet Explorer.*

The Hypermedia Time-based (HyTime) (6) document structuring language is a standard developed to permit human communications in a variety of media and to permit all media technologies to compete against each other in an environment that is capable of supporting all combinations of the media. HyTime was created for the digital publishing industry as a means to integrate all aspects of information collection and representation. It is based on the SGML standard, and consists of several modules. Even though different platforms may have been used to create the information, by standardizing the syntax for documents HyTime brings together people, software, and departments. Representing abstract time dependencies (not addressed in other standards) and hyperlinks are the focus of HyTime, and it is therefore a model that can support any combination of multimedia, hypermedia, time, or space specifications. Furthermore, if the system cannot render some of the objects in the HyTime document, then *blankness* or *darkness* will be rendered to preserve the time/space relationships within the document.

HyTime's addressing capabilities allow the identification of hypermedia reference links from anchors to targets (a chapter of a document, a series of video frames, etc.). The target may be a file outside of the HyTime document and, as such, the model knowns about media types but relies on an application-dependent SGML notation to tell it *how* to access media objects. Referencing elements within the same document is made easier, since each document has its own name space of unique names. Three modes of addressing are supported: by name, by position (in some arbitrary measurable universe or coordinate space), or by semantic construct. Elements can be linked together using different types of links: *independent, property, contextual, aggregate,* and *span.*

Alignment of elements in HyTime is done in terms of bounding boxes called *events* (or grouped events called *event schedules*) that contain references to data. Each event is placed in a finite coordinate space (FCS) with one or more axis relating to some measurement domain (seconds, minutes, etc.) and addressable range (frame, text, etc.). The dimension of the event on each axis is called the *extent,* and the dimension is marked with *quanta.* Absolute as well as relative (by referencing other elements' quanta) temporal specifications can be made, as can delays.

The event projection module maps the FCS of the source into the FCS of the target. In this way, events in a schedule in a source FCS can be first modified (e.g., using "wands") and then projected (e.g., using "batons") onto a target FCS's schedule. Further details on HyTime, including examples, may be found in (7).

Substantial effort has been made by the developers of ScriptX (8) at Kaleida to create a multimedia application platform that handles temporal elements.

VISUAL TECHNIQUES FOR DOCUMENT AUTHORING

It is clear that the potential complexity of multimedia documents dictates, to some extent, the complexity of the authoring tools. This section focuses on several important and significant authoring tools that allow authors graphically to model multiple media and their temporal relationships. The novelty of the graphical representations is emphasized over the particular document architectures, as is the ability of the particular model to represent interactive runtime events as opposed to predetermined absolute events.

General Timeline

Perhaps the most prevalent visual model is the timeline (9), a simple temporal model that aligns all events (start and end events of media objects) on a single axis that represents time. Since the events are all ordered in the way they should be presented, exactly one of the basic point relations, *before* (<), *after* (>), or *simultaneous to* (=), holds between any pair of events on a single timeline (see Fig. 2). Although the timeline model is simple and graphical, it lacks the flexibility to represent relations that are determined interactively at runtime.

For example, assume a graphic (e.g., a mathematical graph) is to be rendered on the screen only until a user action (e.g., a mouse selection) dictates that the next one should begin to be rendered. The start time of the graphic is known at the time of authoring. The end time of the graphic depends on the user action and cannot be known until presentation time; hence the scenario cannot be represented on the traditional timeline, which requires a total specification of all temporal relations between media objects.

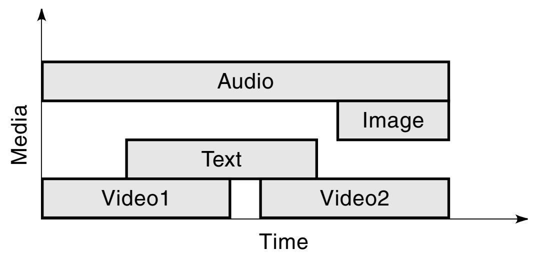


Figure 2. The basic timeline model. Although it is relatively simple, this model visually captures the intramedia relationships and is the basis for several commercial tools.

Some commercial authoring and editing products that use this paradigm are: Macromedia Director, AVID media composer, and Adobe Premiere.

An Enhanced Timeline Model

In Ref. 1, Falchuk, Hirzalla, and Karmouch present a visually and functionally enhanced timeline model that provides additional graphical entities that relay temporal information. In this approach, user actions are modeled as objects on the vertical axis of the timeline (usually reserved for media such as text, graphics, audio, and video). A new type of media object called *choice* is added and is associated with a data structure with several important fields: `user_action`, `region`, and `destination_scenario_pointer.`, `User_action` completely describes what input should be expected from the viewer of the presentation; for instance, `key-press-y` or `left-mouse`. `Region` describes what region of the screen (if applicable) is a part of the action; for instance, `rectangle(100,100,150,180)` may describe a rectangle in which if a user clicks with the mouse, some result is initiated. `Destination_scenario_pointer` is a pointer to some other part of the scenario or a different scenario.

This media object "choice" may be placed directly on the traditional timeline. Suppose there is a scenario in which a video of American presidents is being rendered, along with an audio track. The video serves to introduce us to three American presidents, Clinton, Bush, and Reagan. Suppose again we have rendered text boxes that display the name and the age of each president as they are introduced in the short video clip. Now suppose the authors wish to create some additional timelines, one for each president. In this way a user might make some selection during the playback, the result of which would be a "jump" to a more in-depth presentation of the president currently being introduced (i.e., active multimedia). To do this each of the in-depth scenarios must be authored and then three choice objects must be added to the original timeline. The objects' data structures contain `user_action=left-mouse`, `region=the appropriate layout location on the screen`, and `destination_scenario_pointer=the appropriate scenario`. The choice objects are added in Fig. 3. In effect, the active multimedia scenario is finished. Since the choice data structures are completed, each such object refers to some other subscenario, as in Fig. 3.

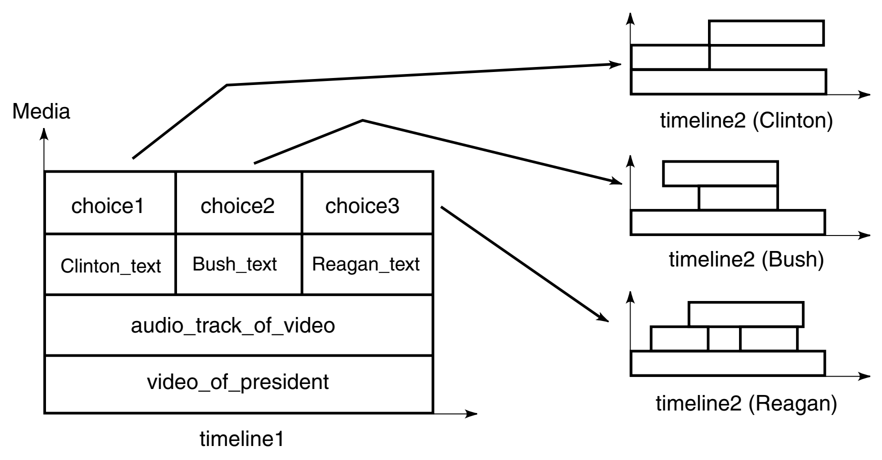


Figure 3. Choice objects in an extended timeline and their associated destinations. Choices represent "windows of opportunity" for asynchronous events.

The resulting scenario is indeed an interactive one. Note that choice objects, like other media objects, have a duration, meaning that the user has a "window of opportunity" to make the action that initiates the choice. If the associated action is not made during this time, the user loses the chance to make it. That is, if the user does not make the mouse selection while the text object `Clinton_text` is being rendered, he or she will continue to see the rendering of timeline1 and a new choice (the one associated with President Bush) will be offered to the user. If the appropriate choice is made, rendering continues from the destination timeline and the original scenario is terminated.

In the enhanced visual model, the traditional rectangle that represents a media element on a timeline is split into three basic units, as shown in Fig. 4(a). The edges of these units, which represent start or end events, are either straight or bent lines. Straight lines represent synchronous events and bent lines represent asynchronous events. The actual time of an asynchronous event can only become known at runtime, and will be shifted by δ seconds to the right on the time axis, where δ is a nonnegative number representing the user response delay (see Fig. 4b). Again though, at authoring time the value of δ is unknown.

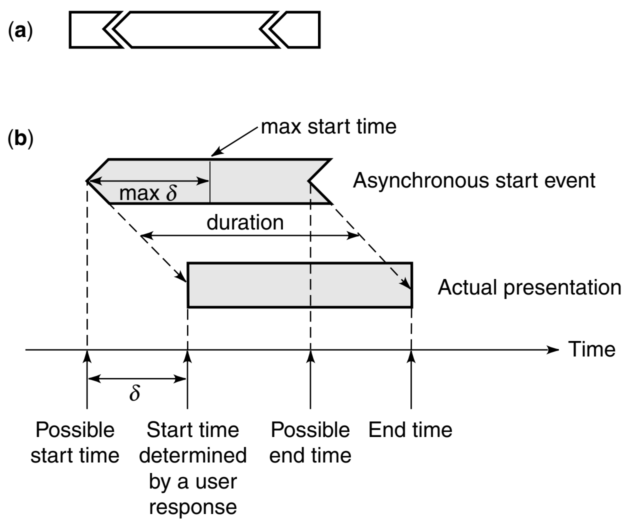


Figure 4. The extended timeline—(a) Basic representation units. (b) User response delay.

The author may specify a limit to the value of δ where, if the user does not respond to some interactive prompt, a default response may be assumed (that starts rendering the media object in the example shown in Fig. 4b). Therefore, defining the maximum value for δ is useful and necessary. The length of the unit from the left sharp point to the right straight line represents the maximum value of δ (see Fig. 4b).

Based on the three basic units shown in Fig. 4(a), many different forms could be used, but only six of them are applicable to interactive scenarios. These six shapes and their descriptions are shown in Fig. 5. The assumptions used in forming the shapes are as follows:

1. An event that is temporally related to an asynchronous event is itself asynchronous. A simple example is that if the start time of a media element that has a specific duration is unknown, then its end time is also unknown.
2. The user response delay, δ, is a nonnegative value.

Petri Nets

In the Petri Net model, media intervals are represented by "places" and relations by "transitions." Petri Nets are a formal graphical method appropriate for modeling systems that have inherent concurrency. The Petri Net model has been used extensively to model network communication and complex systems. A classification scheme exists that partitions Petri Nets into three classes: (1) those with Boolean tokens (places marked by at most one unstructured token), (2) those with integer token (places marked by several unstructured tokens), and (3) those with high-level tokens (places marked by structured tokens with information attached).

As shown in Fig. 6, each of the basic point relations, *before, simultaneous to,* and *after,* can be modeled by a transition in conjunction with a delay place δ. The delay place has a nonnegative value that represents an idle time. If the delay is not known at authoring time, temporal inequalities can be expressed, thus allowing user interactions. However, unlike the timeline model, the graphical nature of the Petri Net model can become complex and difficult to grasp when the document becomes relatively large. Figure 6 shows a possible Petri Net representation of the scenario originally shown in Fig. 2.

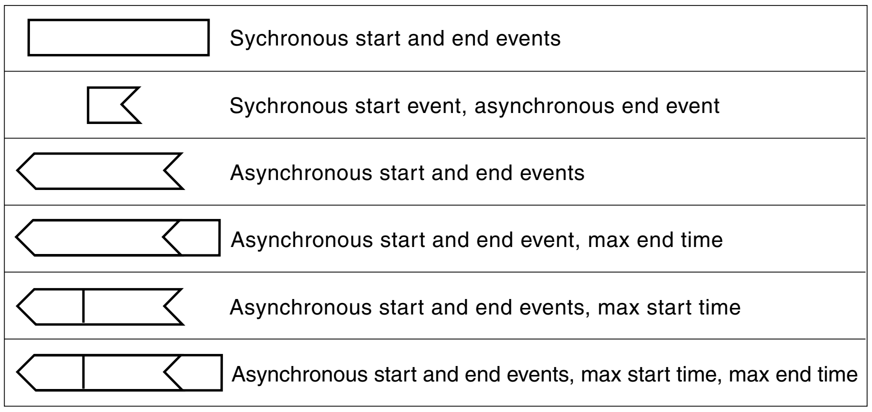

Sychronous start and end events
Sychronous start event, asynchronous end event
Asynchronous start and end events
Asynchronous start and end event, max end time
Asynchronous start and end events, max start time
Asynchronous start and end events, max start time, max end time

Figure 5. Applicable units for interactive multimedia scenarios. These units reasonably capture the relevant event types.

Although this model is often used in research, no significant commercial products have emerged that use it. This reflects the model's lack of practicality in comparison to the timeline model.

MEDIADOC

The logical structure of MEDIADOC (10) uses abstract objects to represent the document in an aggregation hierarchy. Classes include *independent objects, sequential objects, concurrent objects,* and *media objects*. Media object classes have subclasses such as *frame, sound,* and *image*. Other classes, such as *sequential* and *concurrent,* allow specification of temporal relations. MEDIADOC supports the $<$, $=$, and $>$ temporal relations between two objects. These relations are based on events that include time of day, scene start or end, object start or end, and other events. Temporal relations are represented graphically with a unique and effective representation.

Figure 7 illustrates a simple scenario. Circles represent media objects and rectangles represent temporal information. The scene starts (right facing triangle) with graphic G1 commencing immediately (= Start of Scene). G1 endures for 7 s. Five seconds after the termination of G1, T1 is rendered (5 s $>$ G1). The scene then ends (left-facing triangle). MEDIADOC can also support choice graphically as a "splitter" that forms two threads of scenario, but the scenario starts to become unreadable if too many choices are used. To assist in editing, MEDIADOC provides a timeline representation (called a SORT graph) of the scenario in which nondeterminate objects are placed on the graph but shaded with a different texture. Furthermore, scenario verification can be done by the author or automatically by the system.

CMIFed Multimedia Authoring

The CMIFed multimedia authoring (11) provides users with a novel graphical way of visualizing and representing multimedia scenarios. CMIFed offers the traditional timeline-type visualization, called the "channel view" (Fig. 8). Synchronizing media objects can be achieved by using CMIFed "sync-arcs."

As shown in Fig. 8, sync-arcs synchronize the start of the audio track to the end of the logo graphic, as well as synchronizing the start of the video to the start of the audio. The hierarchy view is a novel way of visualizing both the structures of the scenario and the synchronization information using nested boxes. Boxes that are placed top to bottom are executed in sequential order, while those placed in left-to-right order are executed in parallel.

Firefly

Firefly (9) is a powerful system that supports and models synchronous as well as asynchronous behaviors. Each media object is modeled by two connected rectangular nodes representing start and end events. Any other event that would be used for synchronization (such as a frame in a video) is called an internal event and represented by a circular node that is placed between the start and end events. In Firefly, asynchronous events contained in a media item are represented by circular nodes that float above the start event. Temporal equalities between events are represented by labeled edges connecting these events. Figure 9 shows an example in which an image of a car is presented along with background data. The user may select different parts of the car (e.g., the door,

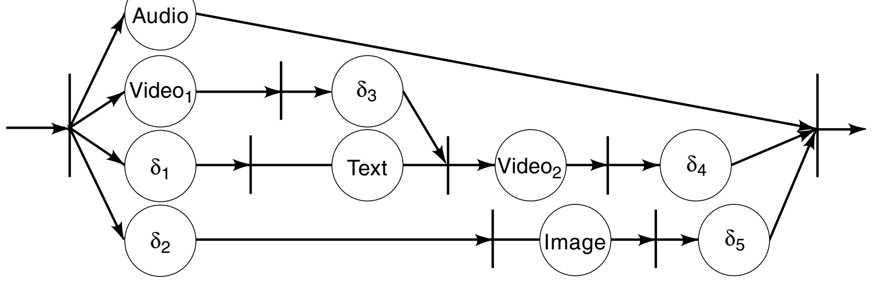


Figure 6. Petri Net model for the scenario shown in Fig. 1.

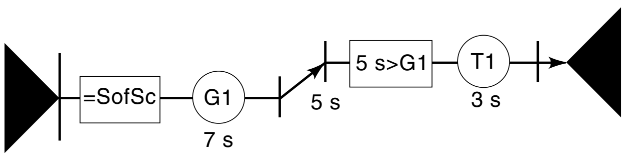


Figure 7. Graphical representation of synchronization specification using MEDIADOC.

the hood) and be presented with a description of that particular part. User selections may or may not be made.

Macromind Director 6.0

Finally, a product by Macromedia Inc. called *Director 6.0* (12) is a very popular suite for creating, editing, and executing multimedia presentations. Director features true objects and drag and drop behavior, and 100 channels for independent graphic elements called "sprites." Furthermore, Director allows instant publishing of interactive multimedia documents onto the World Wide Web and supports streaming media applications that allow bandwidth-intensive media such as video to begin to play immediately on remote machines as opposed to waiting for them to download completely.

Summary

Clearly there are a number of interesting, novel, and effective tools for authoring both passive and active multimedia documents. It can be noted that in almost all graphical authoring tools, the complexity of the visual representation grows with the complexity and nondeterminism of the document being authored. In particular, Petri Nets and Firefly documents tend to become unwieldy as the interactivity of the document is increased. Other methods, such as CMIFed and extended timelines, scale better. The simple timeline is likely the best choice for simple documents. However, if the document must model asynchronous events, the basic timeline model is not satisfactory. Extensions to the timeline model to support these types of events have been investigated.

CURRENT AND FUTURE ISSUES IN AUTHORING SYSTEMS

Current research into the areas of document standards, authoring, and editing is still active. This section first surveys some other important related work not covered in the previous section and then introduces the mobile agent paradigm. This paradigm for data access and interaction may have a profound effect on multimedia documents and authoring systems.

Other Related Work

Recent active areas in related research fields have focused on several key areas. These areas include adding temporal structure to multimedia data, architectures and data models, issues related to time-dependent data, and extensions to standards proposals to provide additional support. Examples of such research proposals follow.

Karmouch and Khalfallah (13) present an architecture and data model for multimedia documents and presentational applications. In this proposal *information objects* are used to model the hierarchic structures of multimedia data, each of which has an associated *presentation activity* that models the rendering process for that object. Composite objects and aggregate objects can be used to model complex multimedia information. Presentational objects synchronize their subordinates using *synchronous message passing*. Communicating synchronous processes (CSPs) and a CSP language provide a mechanism for coordination. Multimedia documents have both logical and layout architectures and associated *computing architectures* that describe the activity of the presentation servers using a graphical notation. A conceptual schema for multimedia documents is proposed using an extension of the Entity-Relationship (ER) model.

Huang and Chu (14) propose an ODA-like multimedia document system. This system takes an object-oriented approach to extending the International Standards Organizations (ISO) ODA to support continuous media as well as static media. The ODA structure is composed of the document profile, structure model, computational features, content architecture, and processing model. This proposal extends ODA such that it may model temporal aspects of continuous media (e.g., video and audio) that have a temporal duration. In this scheme, the control mechanism is described in the view of objects, and behavior properties are also viewed as objects to which end the authors claim improved flexibility over ODA.

Schloss and Wynblatt (15) present a layered multimedia data model (LMDM) consisting of the following layers from the "top" down: Control (CL), Data Presentation (DPL), Data Manipulation (DML), Data Definition (DDL). The DDL allows specification of objects including persistent data or instructions to generate data. The DML allows the grouping of objects into multimedia events within a frame of reference with an *event time*. The DPL describes how data are to be presented to the user. This involves specifying the playback devices, display methods, etc. Multimedia presentations may be reused on different systems since the DPL is system independent. The CL describes how compound presentations are built from one or more other presentations. *Signals* can be accepted from I/O devices and users.

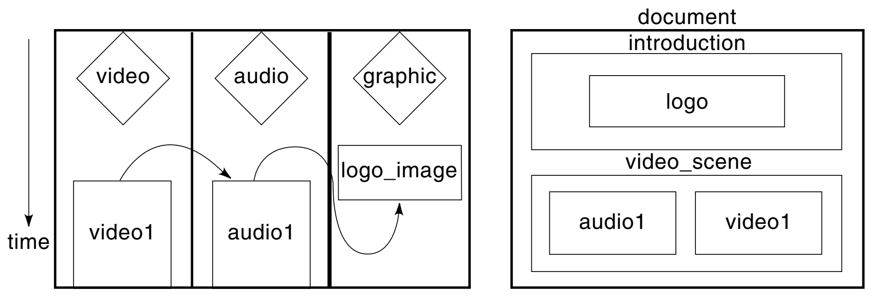


Figure 8. CMIFed channel view (left) and hierarchy view (right) of a temporal document. Sync arrows in the channel view represent synchronization and boxes in the hierarchy view represent parallel rendering.

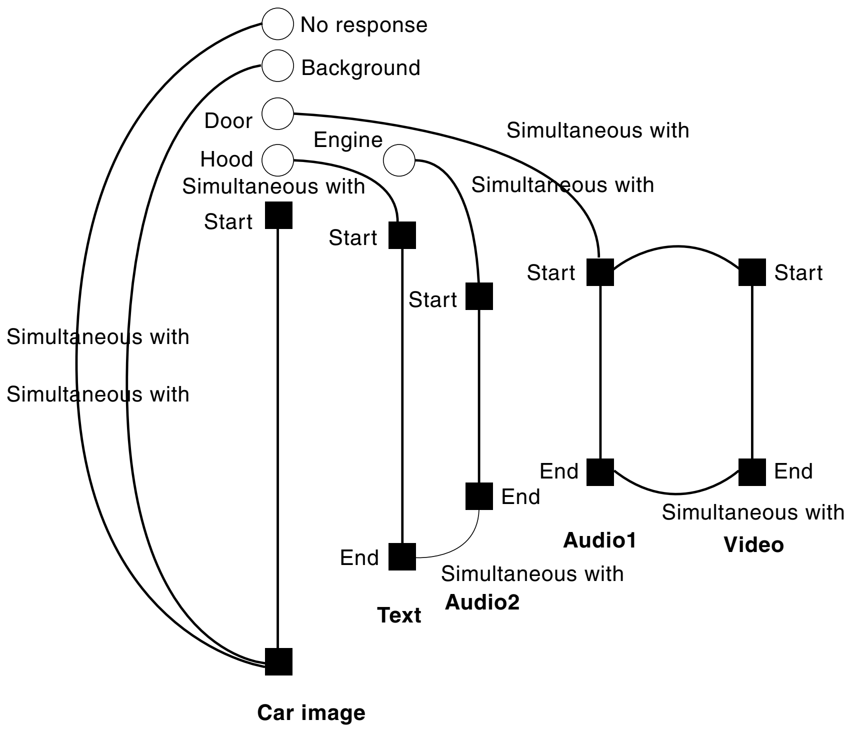


Figure 9. A temporal view of an active multimedia document using Firefly.

Jourdan, Layaida, and Sabry-Ismail (16) present a robust authoring environment called MADEUS that makes use of extended temporal constraint networks and implements a multimedia presentation layer. Finally, Fritzsche (17) presents a granularity independent data model for time-dependent data. This work presents multimedia data more theoretically and abstractly than merely video frames or audio samples.

The Mobile Agent Paradigm and Multimedia Authoring

In the classical client-server model there are two main entities. The *server* is the service provider that typically idles while it waits for well-formed requests to come onto the port that it monitors. The *client* is the service consumer that sends a particular request message to the server when it needs a service performed. The mobile agent-model is somewhat different. A mobile agent can be defined as a program that is able to migrate from node to node on a network under its own control for the purpose of completing a task specified by a user. The agent chooses when and to where it will migrate and returns results and messages in an asynchronous fashion. In other words, mobile agents are sent to, and run beside, the remote data servers, and interaction with the remote data is not limited to the network Application Program Interface (API) otherwise afforded to clients. Mobile agents (see Ref. 18 for a thorough explanation) do not require network connectivity with remote services to interact with them. A network connection is used for a one-shot transmission of data (the agent and possibly its *state* and *cargo*) and then is closed. Agent results in the form of data do not necessarily return to the user using the same communications trajectory, if indeed re-

Table 3. A Comparison of the Client-Server and Mobile Agent-Based Approaches to Document Composition

Client-Server-Based Authoring	Mobile-Agent-Based Authoring
Client requests must come from a fixed set of operations from within the server's API.	Agents are programs and have access to all of the constructs of their particular language (e.g., loops).
Multiple invocations require multiple client requests.	Multiple invocations are iterated at the service itself in local memory.
The client iterates over the network.	Agents run remotely while originating host remains free.
Server decides which data to return to the client.	Agents filter data both locally and remotely.
Data return directly to client.	Agent may jump to any number of intermediate nodes.

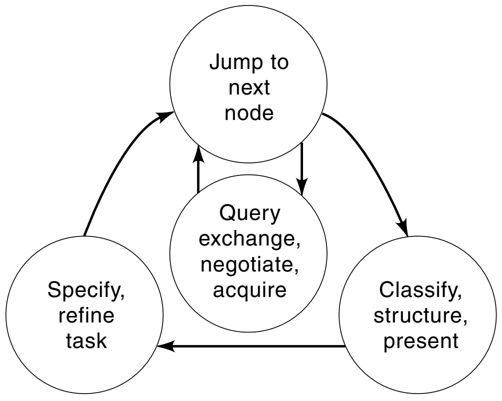

Jump to next node
Query exchange, negotiate, acquire
Specify, refine task
Classify, structure, present

Figure 10. Pseudocode describing the filling in of a document's sections (left), and the general state diagram (right) showing the overall operation of the mobile agent. A mobile agent can also be considered a mobile document, programmed to "fill itself up" with appropriate media.

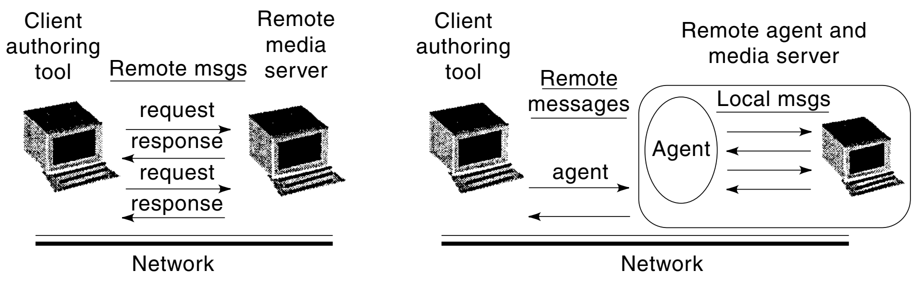


Figure 11. The traditional approach to authoring (left) versus the mobile agent approach (right). The mobile agent approach may reduce network traffic and save the end-user time by physically co-locating with media servers and agents and intelligently gathering media.

sults at this node are gained or even expected. Alternatively, the agent may send itself to another node from the intermediate one taking its partial results from the intermediate node with it. Table 3 illustrates some of the differences from the client-server model.

In general, we can say that the mobile agent model offers several key advantages over the client-server model, including the following: (1) It uses less bandwidth by filtering out irrelevant data (based on user profiles and preferences) at the remote site before the data are sent back, (2) ongoing processing does not require ongoing connectivity, and (3) the model saves computing cycles at the user's computer.

As most document authoring systems, database applications, and legacy systems are client-server-based, the mobile agent paradigm will take time to become widespread. However, research laboratory prototypes (19) and some commercial products (20) have shown promise. Traditional servers will have to be extended with complex semantics to be able to support mobile agents that are clearly much more than merely messages—they have goals, state, cargo, etc. Figure 10 illustrates how a mobile agent, programmed in such a way as to collect media intermittently on certain topics, might be coded to travel from one agent server to another on a network. The agent knows which type of media to collect because it has either (1) been explicitly programmed by the user, or (2) has implicitly learned what the user likes by discovering patterns in the user's day-to-day computing tendencies.

When the mobile agent has finished collecting media to satisfy its mission, it returns to the user's computer and presents the data to the user. The key points are as follows:

- When the agent returns with a document to be presented, both the logical document structure and document content are completely determined by the user's preferences and explicit choices.
- The impact on authoring systems is significant—particularly those that are distributed. In this paradigm, the document authoring process can be thought of as consisting of three parts: (1) specifying the document preferences, (2) dispatching a mobile agent on a distributed system to collect media that can be used in the document, and (3) presenting the results in an arbitrary logical and layout format. This is a large departure from the traditional client-server approach of almost all distributed authoring systems (see Fig. 11).

CONCLUSIONS

This article has introduced the concept of multimedia documents of different kinds and provided definitions of important and commonly used terms in this domain. Several standards that might support multimedia documents were briefly surveyed. Visual approaches to authoring asynchronous multimedia documents were surveyed and then the mobile agent paradigm for authoring documents was introduced and the significance of traditional authoring systems emphasized. Multimedia documents containing heterogeneous media require complex storage, editing, and authoring tools. While there are numerous document standards, few have addressed all of the requirements of documents containing multiple, time-based media. Furthermore, representing document structures and playback scenarios visually is challenging and only a small number of effective tools have emerged in both the commercial and research realms. Authoring systems for these documents must aid in both the spatial and temporal layout of media and may also have to support the modeling of user interactions with time-based media.

ACKNOWLEDGMENTS

We would like to thank our colleagues, Nael Hirzalla, Habib Khalfallah, and James Emery, for their contributions to this article.

BIBLIOGRAPHY

1. B. Falchuk, N. Hirzalla, and A. Karmouch, A temporal model for interactive multimedia scenarios, *IEEE Multimedia,* Fall 1995, 24–31.
2. M. Buchanan and P. Zellweger, Specifying temporal behavior in hypermedia documents. *Proceedings of the ACM Conference on Hypertext,* New York: ACM Press, Dec. 1992, pp. 262–271.
3. ISO 8879, Information Processing Text and Office Systems—Standard Generalized Markup Language, 1986.
4. ISO 8613-1 (CCITT T.411), Open document architecture and interchange format—introduction and general principles, 1988.
5. T. Berners-Lee and D. Connolly, Hypertext Markup Language—2.0, IETF HTML Working Group, http://www.cs.tu-berlin.de/~jutta/ht/draft-ietf-html-spec-01.html.
6. ISO/IEC DIS 10744, Information Technology—Hypermedia/Time-based Structuring Language (HyTime), August 1992.
7. R. Erfle, Specification of temporal constraints in multimedia documents using HyTime, *Electronic Publishing,* **6** (4): 397–411, December 1993.
8. R. Valdes, Introducing ScriptX, *Software Tools for the Professional Programmers,* **19** (13): November 1994.
9. G. Blakowski, J. Huebel, and U. Langrehr, Tools for specifying and executing synchronized multimedia presentations. *2nd Int. Workshop on Network and Operating System Support for Digital Audio and Video,* Nov. 1991.

10. A. Karmouch and J. Emery, A playback schedule model for multimedia documents, IEEE Multimedia, Spring 1996, 50–61.
11. R. Rossum et al., CMIFed: A presentation environment for portable hypermedia documents, *Proc. of ACM Multimedia '93,* New York: ACM Press, August 1993, pp. 183–188.
12. Macromedia Inc., Macromind Director 6.0, http://www.macromedia.com.
13. H. Khalfallah and A. Karmouch, An architecture and a data model for integrated multimedia documents and presentational applications, *Multimedia Systems,* **3**: 238–250, 1995.
14. C. Huang and Y. Chu, An ODA-like multimedia document system, *Software—Practice and Experience,* **26** (10): 1097–1126, October 1996
15. G. Schloss and M. Wynblatt, Providing definition and temporal structure for multimedia data, *Multimedia Systems,* **3**: 264–277, 1995.
16. M. Jourdan, N. Layaida, and L. Sabry-Ismail, Presentation services in MADEUS: An authoring environment for multimedia documents, INRIA Research report no. 2983.
17. J. C. Fritzsche, Continuous media described by time-dependent data, *Multimedia Systems,* **3**: 278–285, 1995.
18. Special Issue on Agents, *Communications of the ACM,* **37** (7): 1994.
19. B. Ford and A. Karmouch, An architectural model for mobile agent-based multimedia applications, *Proc. of CCBR'97,* Ottawa, Canada, April 1997.
20. J. E. White, Telescript Technology: The Foundation for the Electronic Marketplace, General Magic White Paper, General Magic, 1994.

BENJAMIN FALCHUK
AHMED KARMOUCH
University of Ottawa

AUTOMATA. See AUTOMATA THEORY.
AUTOMATA, CELLULAR. See CELLULAR AUTOMATA.

AUTOMATA THEORY

AUTOMATA AS MODELS FOR COMPUTATION