Perspectives on Multimedia Synchronization: Giving Control to Authors

M. Cecelia Buchanan

Washington State University
School of Electrical Engineering and Computer Science
Pullman, WA 99164-2752
buchanan@eecs.wsu.edu

1. Introduction

The prevailing definition of “multimedia synchronization” pegs it as a network and operating system level issue that deals with how to deliver streams of multimedia data in a way that maintains a specific set of temporal relationships. A typical example of a multimedia synchronization problem is ensuring that video and audio streams are delivered to an application in a way that maintains lip synchronization between the media streams. Research in this field has been quite active, and a number of algorithms have been (and are continuing to be) developed to address this issue.

My research approaches the problem of multimedia synchronization from a different point of view, namely that of a multimedia author. In particular, most of my work has focused on how authors specify the intended temporal relationships among media streams. Section 2 provides a brief overview of my research, and Section 3 discusses my perspective the unsolved problems in this area.

2. Specifying Temporal Behavior in Multimedia Documents

To date, my research on multimedia synchronization has been based on the hypothesis that incorporating high-level semantics into a multimedia system's temporal specification language can significantly simplify the creation and maintenance of multimedia documents [1-5]. My research has identified two major advantages of this approach. First, it simplifies the maintenance of the resulting documents by maintaining an explicit record of the intended temporal relationships. Second, it allows temporal specifications to be processed by an automatic *temporal formatter*, which can produce temporal layouts for multimedia documents much as a spatial formatter, such as Microsoft Word or $\LaTeX$, automatically produces spatial layouts for traditional documents. Furthermore, temporal formatters simplify document creation and maintenance by relieving authors from performing many of the tedious and error-prone tasks associated with creating multimedia documents.

To make the construction of automatic temporal formatters feasible, I had to address two questions. First, what sort of temporal specification language is made available to authors to allow them to provide the system with sufficient information to automatically generate temporal layouts? Second, how can the system construct a temporal layout that best satisfies the temporal relationships specified by the author? The results of my research show that a temporal specification based on temporal constraints supports these mechanisms well. The remainder of this section discusses the four novel contributions of my work.

My first contribution is the design of an improved temporal specification language, based on temporal constraints, that significantly reduces the need for authors to manipulate timings when specifying the temporal behavior of multimedia documents. This language increases the expressiveness of multimedia documents because it can describe both predictable and unpredictable behaviors. In addition, the characteristics of this language allow it to be processed by an automatic temporal formatter, such as the one outlined below. Specifically, it represents the intended temporal relationships among the media components explicitly, and it allows authors and media managers to specify the flexibility parameters required by the formatter, namely the ability to stretch and shrink media component durations.

My second contribution is the design of an algorithm that automatically produces temporal layouts for temporal specifications containing media components with predictable and unpredictable behavior. This formatter uses a two-phase approach.

The first phase of the formatter, called the *scheduler*, preprocesses the temporal specification *before* presenting the document. It uses a constraint satisfaction algorithm, based on linear programming, to lay out events and media components in time so as to conform to the author's temporal specification. The scheduler may stretch and shrink the durations of media components to produce an ``optimal'' *partial layout* in which it assigns times to the predictable portions of the temporal specification, while leaving those for the unpredictable portions unresolved. The scheduler's ability to stretch and shrink media component durations is conceptually similar to $\TeX$'s ability to stretch or shrink the amount of white space (glue) between the items to be typeset (boxes) [6]. By associating *costs* with stretching and shrinking the durations of media components, authors can control the way in which the scheduler uses this flexibility.

The second phase of the formatter, called the *runtime system*, operates while the document is being presented. It completes a temporal layout by merging the predictable and unpredictable portions of the partial layout in response to the occurrence of unpredictable behavior, such as user interaction and program states.

The third contribution of my research is the design of a complete architecture for a multimedia document system, consisting of direct manipulation editors, an automatic temporal formatter, and a presentation system. To test this architecture, I implemented a full prototype system called Firefly and used it to construct a variety of sample multimedia documents. The results of this experiment verified that the temporal formatting algorithm described above has acceptable performance.

Finally, based on an analysis of how Firefly's solution fits in with those supplied by other concurrently developed systems, I co-developed a framework for understanding automatic temporal formatters and exploring the issues surrounding them. This framework unifies Firefly's strengths in handling predictable and unpredictable temporal behavior, and attempting to correct temporal mismatches *before* presenting the document, with other systems' abilities to present multimedia documents in heterogeneous environments and to account for media delays. This framework allows the designers of temporal formatters to evaluate and to select the set of characteristics best-suited to the types of documents they intend to handle. In addition, it can be used to classify, evaluate, and compare new formatters as they become available.

3. Future Areas of Research

Research done to date has addressed only a handful of the issues related to providing authors control over multimedia synchronization. Future work in this area needs to address problem in three broad areas: (1) developing improved user interfaces for specifying temporal behavior in multimedia documents, (2) devising additional authoring tools to automate the creation of multimedia documents, and designing temporal models that will allow these tools to represent and reason about time, and (3) designing protocols to support communication between the author's media synchronization mechanisms and those at the network and operating system layers. The remainder of this section briefly outlines some of the issues surrounding these problems.

3.1 User Interfaces

One of the biggest obstacles that must be overcome is developing a visual representation that authors can use to specify the temporal behavior of multimedia documents. Currently, most commercially-available multimedia authoring systems provide user interfaces based on timelines. Although timelines can be used to represent simple types of documents with predictable behavior, such as synchronizing and audio recording with a set of slides, they are not powerful enough to represent more complex unpredictable behavior, such as a user pressing a button or the calendar tool notifying the user that an interesting talk is about to take place. Instead, authors must resort to scripting languages to specify this type of behavior. For example, an author could create a hypertext dictionary by associating a script with the text object containing the terms. This script would identify the selected term, look it up in the dictionary, and display the corresponding definition. Similar scripts could be associated with other media objects to produce the other forms of asynchronous behavior contained in the preceding document. Unfortunately, most multimedia authors are not programmers, and they dislike writing scripts. In fact, one of the biggest complaints of most multimedia authors is that they must have programming skills to take full advantage of the features offered by most multimedia authoring systems.

The user interface for the initial Firefly prototype provided a simple visual user interface in which multimedia documents were represented as graphs. Graph nodes represent events, which are potential points of synchronization within a media object. Graph edges represent the temporal ordering of events, including durations imposed by the media objects and temporal constraints imposed by the author. The length of a graph edge is proportional to the nominal amount of time that should elapse between the occurrence of the connected events. Unfortunately, this notation does not visually represent all of the characteristics of media objects, such as the stretchability and shrinkability of media component durations. In addition, this notation uses graphs, which many authors find to be intimidating.

The author's current research focuses on exploring alternative ways of visually representing the temporal behavior of multimedia documents.

3.2 Authoring Tools and Temporal Models

Authoring tools can help simplify the creation and maintenance of multimedia documents by automating tedious and errorprone tasks. This section briefly outlines the issues involved in developing two authoring tools: one capable of handling environmental mismatches and the other capable of incrementally creating temporal layouts.

3.2.1 Handling Environmental Mismatches

Environmental mismatches arise when the target computing environment lacks the resources and/ or the performance characteristics required by a document. For example, a document containing video might be presented on an ASCII terminal lacking video hardware, or a document might be retrieved from a network via a low-speed modem incapable of handling the required data rates. Environmental mismatches are a concern for authors because they adversely affect a document's presentation, resulting in readers perceiving something other than what the author intended.

Unfortunately, multimedia systems currently provide little support, if any, for authoring and presenting documents in heterogeneous and cross-platform environments. Instead, authors must rely on manual techniques, such as creating multiple versions of their documents. We can simplify the creation and maintenance of this type of document by adding a mechanism for automatically detecting and correcting environmental mismatches. To make this extension possible, four important issues must be addressed.

First, the system needs a way of characterizing the target environment in which a document will be presented. This environment specification must describe all of the computing and network resources that may be required by a document, and it must describe the types of available resources and the ways in which these resources can be used in parallel. For example, a multimedia workstation might have facilities for presenting video with the restriction that only two video windows can be active simultaneously. In addition, the environment specification should be represented in a way that facilitates environmental mismatch detection and corrections algorithms.

To detect environmental mismatches, systems require a characterization of a document's resource and performance requirements as a function of time, and of the target environment's network and hardware resources. An environmental mismatch detection algorithm compares these requirements and identifies any conflicts. Unfortunately, the presence of unpredictable temporal behavior, such as user interaction and programs with unknown durations, complicates environmental mismatch detection, because systems cannot predict when, and in what combination, this behavior will occur. The most promising solution appears to be modifying the runtime formatter to check environmental requirements whenever unpredictable behavior occurs.

Detecting environmental mismatches is only part of the problem. To avoid forcing authors to manually handle environmental mismatches, systems should provide a mechanism for modifying documents so they can be presented in the target environment. Designers should avoid primitive, non-adaptive techniques, such as issuing an error message indicating that the document cannot be presented due to environmental problems, and presenting partial documents that omit the media segments not supported by the environment. Instead, they should focus on adaptive algorithms that can substitute alternative media segments that the target environment can handle. For example, to present a document containing video on an ASCII terminal, a system could replace the video with a textual description of it.

Permitting systems to adapt documents to their target environments complicates the process of creating and maintaining documents, because authors must specify alternative media segments and the intended temporal relationships among them. Therefore, it will also be necessary to develop tools to help authors manage this complexity.

3.2.1 Providing WYSIWYG Editors

Researchers also need to develop WYSIWYG (What-You-See-Is-What-You-Get) multimedia document editors that incrementally produce temporal and spatial layouts for multimedia documents in response to an author's actions in the editor. Conceptually, these editors would be the multimedia equivalent of traditional WYSIWYG document editors such as Microsoft Word, Aldus PageMaker, and FrameMaker. Because WYSIWYG editors allow authors to see exactly how each modification affects the a document's temporal and spatial layout, they should simplify document creation and maintenance as well as help authors locate errors in the specifications.

Many issues must be addressed to make the development of WYSIWYG multimedia document editors possible. First, document editors need improved paradigms that allow authors to manage the complexity of specifying both a document's intended spatial and temporal behavior. The most promising solution appears to be the approach used in MODE in which authors are given multiple views of a document, allowing them to focus on each of the different aspects of a specification (e.g., temporal behavior and screen layout) in isolation. Alternative representations make the whole problem even more complicated. In addition, WYSIWYG editors will also require an incremental temporal formatter capable of incrementally updating a document's temporal layout in response to an author's actions in the editor.

3.3 Communicating with Media Synchronization Mechanisms at the Network and Operating System Layers

Multimedia authoring systems and temporal formatters currently do not take advantage of the network and operating system level media synchronization and resource allocation mechanisms being introduced by other researchers. However, researchers must begin to explore ways of interfacing these components if they hope to find robust solutions to the problems of environmental mismatches and unpredictable media delays. Methods must be found to allow temporal formatters to communicate a document's resource and time requirements to the lower level mechanisms, which can determine if the required (hardware and software) resources are currently available. If the document's requirements cannot be satisfied, the lower level mechanisms must negotiate with the formatter to adapt the document to the current environment.

References

[1] Buchanan, M. Cecelia and Zellweger, Polle T., “Scheduling Multimedia Documents Using Temporal Constraints,” *Network and Operating System Support for Digital Audio and Video*, P. Venkat Rangan (Ed.), Springer-Verlag, Berlin Heidelberg New York, 1993, pp. 237-249.

[2] Buchanan, M. Cecelia and Zellweger, Polle T., “Specifying Temporal Behavior in Hypermedia Documents,” *Proceedings of the ACM Conference on Hypertext (ECHT'92)*, Milan, Italy, December 1992, pp. 262-271.

[3] Buchanan, M. Cecelia, Zellweger, Polle T., and Pier K., “Multimedia Documents as User Interfaces” (videotape), *INTERCHI'93*, Amsterdam, The Netherlands, April 1993.

[4] Buchanan, M. Cecelia and Zellweger, Polle T., “Automatic Temporal Layout Mechanisms,” *Proceedings of ACM Multimedia'93*, Anaheim, CA, August 1993, pp. 341-350.

[5] Buchanan, M. Cecelia and Zellweger, Polle T., “Automatically Generating Consistent Schedules for Multimedia Documents,” *Multimedia Systems Journal*, 1(2), September 1993, pp. 55-67.

[6] Knuth, D. and Plass, M., “Breaking Paragraphs into Lines, “*Software-Practice and Experience*, 11(11), pp. 1119-1184.