

# A Flexible Architecture for Heterogeneous Replayable Workspaces

Nelson R. Manohar and Atul Prakash  
Department of Electrical Engineering and Computer Science  
University of Michigan, Ann Arbor, MI 48109-2122 USA.  
email: {nelsonr,aparakash}@eecs.umich.edu

## Abstract

*This paper presents a novel and flexible architecture to support the asynchronous sharing of computer-supported workspaces (CSWs), or simply replayable workspaces. Such workspaces are composed of multiple, interacting tools. Through the capture, re-execution, and manipulation of a session with a CSW, it is possible to reuse valuable collaborative information, (e.g., the how-to process). The session is encapsulated into a session object — composed of heterogeneous media streams that represent input sequences to CSW tools. CSW tools are modeled as plug-in components to the CSW. A stream controller process extends temporal-awareness to a CSW tool and its media. A session manager coordinates the various stream controllers and their media. Our architecture provides flexible coordination of the various tools found on a CSW and fine-grained integration of their heterogeneous media.*

## 1. Introduction

Suppose that while interacting with multiple applications on your desktop, you reach an interesting result. You are likely to document these results and procedures so maybe, *later on*, you can share these — results and procedures — with your colleagues. Some choose to write scripts detailing the steps and results taken. Others, capture these tasks with either analog or digital video. Both approaches are often inadequate for collaborative work due to the potential loss of collaboration content.

Our approach addresses the above problem by supporting the *asynchronous sharing* of interactive application workspaces through the use of **session objects** [9] (see Fig. 1). In this paper, we extend this support to computer-supported workspaces (herein CSW) — composed of multiple tools as opposed to a monolithic application workspace. Such

asynchronously-shared CSWs are referred herein as **replayable workspaces**<sup>1</sup>.

In this paper, we present an architecture for the support of such replayable workspaces. Its key goals are:

**extensibility:** support for multiple applications (CSW tools). It should be possible to record and playback multiple applications, accompanied with temporal annotations such as audio and video.

**modularity:** independence of tools. It should be possible to *plug-in* a tool (e.g., audio record/playback) without affecting other CSW tools.

**reusability:** limited impact to existing tools. It is desirable to reuse existing multimedia tools, such as audio, video, and interface tools.

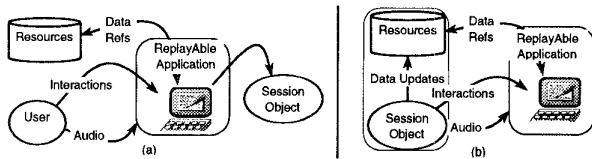
**collaboration:** support for collaborative features such as playback and annotation of session objects [9].

The rest of this short paper is organized as follows. First, we state our motivations for our research on replayable workspaces. Second, we outline the problems that need to be addressed so as to support a replayable workspace. Then, we present an architecture to address those issues. Then, we discuss related work. Finally, we make our concluding remarks.

## 2. Motivation

Our long-term research focuses on the development of toolkits to support collaboration paradigms such as the asynchronous sharing of CSWs. This research is partially motivated by the needs of two projects at the University of Michigan: UARC [4] and the Medical Collab (MDC). Specifically, in the MDC, our goal is the asynchronous sharing of the CSW of a radiologist. The

<sup>1</sup>— although similar to the term *shareable* workspaces found in synchronous collaborative systems, the term *replayable* highlights the asynchronous (later-time use) nature of our research.



**Figure 1. High level view of capture and replay of an interactive session with a replayable application. During capture, inputs to an application, audio annotations, and resource references are recorded into a session object (Fig. 1a). During replay, these are made accessible to the application (Fig. 1b). This paper outlines an architecture to extends support to workspaces composed of multiple applications.**

sharing of such CSW<sup>2</sup> is valuable to radiologists (e.g., for intern training), to clinicians (e.g., for consultation and analysis), and to administrators (e.g., as active documentation). From a collaborative viewpoint, our goals wrt a CSW are to: (1) *reproduce intra-task content*, e.g., at a later time, review the steps taken that led to a diagnosis; (2) *review the task*, e.g., at a later time, browse/annotate such session; and (3) *reuse the task's results and procedures*, i.e., build upon the results contained in the workspace.

### 3. Computer-supported workspaces

In this section, we outline fundamental issues on the asynchronous use of CSWs. A CSW is composed of multiple tools. A *CSW tool* is a user-level process that renders services to the CSW. A *service* is a user-oriented function provided to the CSW. A tool service (say *V*) can either be: (1) triggered<sup>3</sup> by a service (say *W*) or (2) cross-referenced<sup>4</sup> by another (say *A*). For example, our prototype CSW is composed of three CSW tools (and their services): (1) an audio tool (records and replays audio), (2) a video tool (records and replays video), (3) an X-Windows multiplexer (re-distributes the X display stream). Although each tool has some degree of temporal-awareness, the composite CSW does not. Thus, to achieve a properly functioning CSW a later time, we must address the following:

**tool coordination problem:** coordinate the services provided by the various tools found in a replayable

<sup>2</sup>— Such workspaces may consists of tools to: (1) analyze x-rays images, (2) attach audio annotations, (3) analyze MRI scans, and (4) review textual reports.

<sup>3</sup>— e.g., a causal interaction ( $W \leftarrow V$ ) between services.

<sup>4</sup>— e.g., a concurrency interaction ( $A * V$ ) between services.

workspace. For example, it should be feasible to reproduce both causal and concurrent interactions between the services of the tools in the workspace.

**media integration problem:** satisfy temporal constraints between the media handled by tools. For example, it should be possible to synchronize the presentation of inputs to re-execution devices.

In addressing the tool coordination and media integration problems, we must consider how services are to be mapped to tools. There are two basic approaches. In the first approach, the services of multiple tools are integrated into a monolithic-application — using carefully orchestrated, cooperating threads. Our previous prototype followed this approach [8]. This approach delivers complete control over the tool coordination and media integration problems since, at the thread level, fine-grained scheduling and synchronization control is possible. Unfortunately, this low-level control also limits the flexibility of the resulting infrastructure.

In the second approach, each CSW tool remains a user-level process in the replayable workspace. However, the coordination of tool services is now handled by an intermediary process — in our case, referred to as the session manager, such as in *publish-and-subscribe* and message-bus systems. Although the underlying causal model used by these systems can address our tool coordination problem, it can not effectively address our media integration problem which requires support of fine-grained temporal inter-media relationships. Next, we present an architecture that provides modular support of both tool coordination and media integration.

### 4. The replayable workspace

A CSW is composed of multiple tools. A tool is incorporated into the replayable workspace by means of a stream controller, which extends temporal-awareness to tool services and media. Temporal-awareness facilitates the integration of a tool into the replayable workspace, so that use of the tool at a later time is possible. The stream controller abstraction has three components: (1) *controller* — (delivers temporal-awareness to its CSW tool), (2) *temporal media stream* — (herein *tms*, represents a temporal ordering of the data inputs to a tool and its services), and (3) the *CSW tool* itself. A process, referred to as the session manager, coordinates the various stream controllers. Fig. 2 illustrates our replayable workspace architecture, based on the second approach.

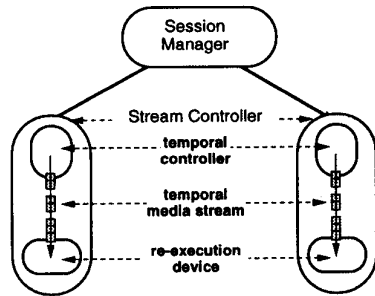


Figure 2. The architecture of the replayable CSW. The CSW is composed of multiple tools. A tool is incorporated into the replayable workspace by means of a stream controller, which extends temporal-awareness to a tool and its media. The stream controller abstraction has three components: (1) controller (delivers temporal-awareness to a tool), (2) temporal media stream (data inputs to a tool), and (3) CSW tool. A session manager process coordinates the various stream controllers.

#### 4.1. The stream controller abstraction

A stream controller delivers temporal awareness to its CSW tool through two abstractions: *media access* and *media control* (Fig. 3). Media access provides low-level device primitives for media handling. Media control provides: (1) intelligence to invoke media access primitives in such a way so as to extend temporal-awareness to the re-execution device and (2) interfaces to media-dependent functions such as adaptive scheduling, performance monitoring, and prefetching.

#### 4.2. The temporal media stream (tms)

A temporal media stream models the inputs to a tool as a temporally-ordered sequence of the following:

**events:** represent units of re-execution, and as such are defined by their re-execution device model.

**states:** Events represent deltas applied over application state and may be statefully dependent on previous events, state, or both. State data provides the basis for a restartable process formulation that allows re-execution at an arbitrary time-index.

#### 4.3. The intra-tool interfaces

In order to manipulate a CSW tool, a controller is required to interface to its tool. Event interfaces allow

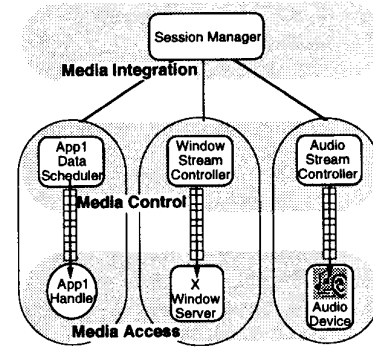


Figure 3. Media handling in the architecture. The session manager focuses on media integration whereas stream controllers focus on media control and access. Media access consists of device primitives for media I/O. Media control provides intelligence to those primitives in such a way so as to extend a homogeneous temporal-awareness model to re-execution devices.

a controller to capture and dispatch events to its tool. State interfaces allow a controller to capture and install state into its tool.

#### 4.4. The re-execution device

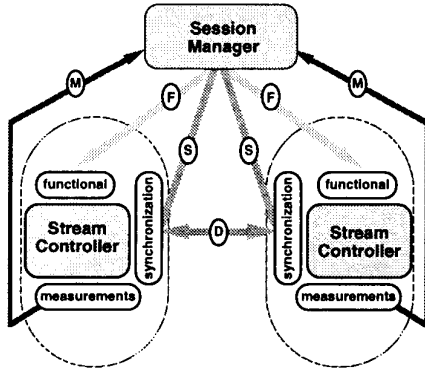
The underlying re-execution device (i.e., typically, a media device or a tool/application) is modeled as a consumer/producer of *tms* events.

#### 4.5. The session manager

The session manager is a user-level process that coordinates the services of the various stream controllers. The session manager allows a user to manipulate session objects through simple *VCR-like commands* — regardless of the tools found on its CSW. Thus, a flexible CSW requires the session manager to be decoupled of:

- tools found in the CSW (i.e., tool coordination).
- *tms* handled by these tools (media integration).

To decouple the session manager from the heterogeneous tools found on its workspace, a homogeneous interface is specified between the session manager and its tools. To facilitate compliance to our interfaces, the stream controller abstraction is provided. Stream controllers enforces a homogeneous interface to media-dependent tool services. For example, a request to *playback a session* gets broadcast to stream controllers as the abstract command: “*replay stream*”.



**Figure 4. Interfaces of the architecture. The architecture uses a client – server model. The session manager coordinates multiple CSW tools — each managed by a stream controller. A stream controller facilitates compliance of a CSW tool to the interfaces of our replayable workspaces: (1) functional interface  $F$  (a homogeneous interface to tool coordination), (2) synchronization interface  $S, D$  (primitives for inter-media integration), and (3) feedback interface  $M$  (evaluation of the infrastructure).**

To decouple the session manager from the heterogeneous media handled by tools on its workspace, its interfaces are specified over a media-independent stream type — i.e., polymorphism. Media-independent commands are dispatched from the session manager to the various stream controllers — (for media-dependent processing such as: (1) capture, representation, and re-execution of its *tms*; (2) scheduling and synchronization; and (3) measurements collection). The implementation of media-dependent functions is thus transparent to the session manager.

#### 4.6. Inter-tool Interfaces

To support tool coordination and media integration, every stream controller must comply with three inter-process interfaces *wrt* the session manager (Fig. 4):

**functional interface (F):** supports tool coordination — its primitives provide abstract machines that build the features of session objects.

**synchronization interface (S,D):** supports media integration — its primitives support fine-grained inter-media relationships between *tms*.

**feedback interface (M):** supports the evaluation of the efficiency of the above interfaces.

When a stream controller complies with our interface specifications, its corresponding CSW tool is said to be **replay-aware**. A replay-aware tool is modeled as a temporally-aware plug-in component to our replayable workspaces. Among other things, it is able to record and re-execute its services and media.

#### 4.7. Transport layer

To preserve the openness and scalability of the architecture, we also specify the inter-process communication (IPC) paths between session manager and stream controllers. A forward-channel  $F(str)$  links session manager to a stream controller  $str$  whereas a back-channel  $B(str)$  links  $str$  to its session manager. Since our interfaces are mapped to these IPC paths, channels need to be at least reliable FIFO point-to-point links. By decoupling interfaces from message delivery, we remove assumptions that a transport layer may impose — for example, we make no assumption about delivery mechanisms such as multicasting between session manager and stream controllers.

#### 4.8. The session object

A flexible workspace requires decoupling the logical representation of a session object from the physical representation of any of its *tms*.<sup>5</sup> Thus, a two level data model is used for our session objects:

**intra-tool data model:** the physical layout of a *tms*. Needs to, and thus only known, to its stream controller.

**inter-tool data model:** the logical representation of a CSW session (meta-data). Needs to be agreed by all streams controllers, so as to enforce homogeneous manipulations over a CSW session.

Finally, session objects are transportable objects. Its persistent image is rooted as a file hierarchy of media directories. The session manager remains unaware of stream-dependent details such as naming scheme, data locality, re-execution device, or layout of stream controller repositories. The session object is thus copied and moved as any other directory.

### 5. Related work

Systems such as XTV [1], and CECED [5], in principle, allow replay of unmodified applications. However, these systems lack media integration mechanisms and interacting with the underlying CSW is not possible.

<sup>5</sup>— Otherwise, functionality is then limited by any of its *tms*.

Although our research seems similar to multimedia authoring<sup>6</sup> [2, 3], the problems are significantly different. Multimedia authoring<sup>7</sup> centers around the *manual crafting* of interactive documents (typically, through temporal scripts or flowcharts). On the other hand, our artifact generation centers around the user-transparent capture of CSW sessions.

Continuous media servers rely on a tightly-coupled media/synchronization server [6, 10] for fine-grained synchronization of pre-selected media. Such functionality coupling is inflexible since it results in a media-dependent approach to media-integration. Our approach, on the other hand, removes awareness of *tms* types from the synchronization server — by delegating media handling to stream controllers.

Finally, as in [7], network-orchestrated playback is also possible — by configuring the session manager to remotely control multiple synchronous CSW sessions on different workstations. However, our orchestration will incur negligible bandwidth requirements (i.e., after startup costs) since network traffic would consist of just commands messages (and not media).

## 6. Concluding remarks

In this paper, we outlined flexible support for the asynchronous (later-time) use of CSWs. Such support addressed: (1) coordination of CSW tool services and (2) integration of the heterogeneous *tms* of these tools. Our novel architecture facilitates handling of media integration and tool coordination, as follows:

- *Removes awareness of media types* from its media integration — (through semantical manipulations over a media-independent type).
- *Encapsulates stream-dependent processing* (such as file access, thread switching, and scheduling) close to the re-execution device and away from media integration.
- *Models tools as plug-in components* — by baselining tool coordination and media integration.

To demonstrate our replayable workspace ideas, we used the University Hospital of Geneva's OSIRIS II tool<sup>8</sup>. Our prototype<sup>9</sup> represents a black box that extends replay-awareness to an Osiris session by coordinating audio, video, and window streams. Finally, we

<sup>6</sup>— e.g., stored media integration, interactive browsing.

<sup>7</sup>— e.g., balancing data intensive segments (e.g., movie clips) to less demanding segments (e.g., text-based interactions).

<sup>8</sup>— at <http://expasy.hcuge.ch/www/UIN/osiris.html>.

<sup>9</sup>— at <http://www.eecs.umich.edu/~nelsonr/systems.html>.

are working on modular mechanisms for the integration of application classes. Our application data controller will represent the gateway to the modular integration and reuse of applications into replayable workspaces.

## 7. Acknowledgments

Our thanks to our proficient programmer, Todd Gee, who has efficiently delivered these ideas into various tools of the replayable workspace. In particular, through analysis of redundancies found on the window stream, Todd obtained a large (lossy) compression ratio during the capture of display-updates, thus dramatically reduced the storage requirements of resulting session objects. Our work has been supported in part by NFS Grant ECS-94-22701, by NFS Cooperative Agreement IRI-92-16848, and by the Rackham Merit Fellowship.

## References

- [1] H. Abdel-Wahab, S. Guan, and J. Nievergelt. Shared workspaces for group collaboration: An experiment using Internet and Unix inter-process communication. *IEEE Communications Magazine*, pages 10–16, November 1988.
- [2] D. Bulterman, G. van Rossum, and et. al. A structure for transportable, dynamic multimedia documents. In *Proc. of the Summer 1991 USENIX Conference*, pages 137–154, Nashville, TN, USA., June 1991.
- [3] M. Cecelia-Buchanan and P. Zellweger. Scheduling multimedia documents using temporal constraints. In *Proc. of the 3rd Int'l Workshop on NOSDAV*, pages 237–249, La Jolla, CA, USA, November 1992.
- [4] C. Clauer and et. al. A new project to support scientific collaboration electronically. *EOS Transactions on American Geophysical Union*, 75, June 28 1994.
- [5] E. Craighill and et. al. CECED: A system for informal multimedia collaboration. In *Proc. of ACM Multimedia '93*, pages 436–446, CA, USA, August 1993.
- [6] R. Dannenberg, T. Neuendorffer, and et. al. Tactus: toolkit-level support for synchronized interactive multimedia. *Multimedia Systems*, 1(1):77–86, 1 1993.
- [7] L. Li, A. Karmouch, and N. Georganas. Multimedia teleorchestra with independent sources: Part 1 — temporal modeling of collaborative multimedia scenarios. *Multimedia Systems*, 1(2):143–153, 1 1994.
- [8] N. Manohar and et. al. Dealing with synchronization and timing variability in the playback of interactive session recordings. In *Proc. of ACM Multimedia '95*, pages 45–56, San Francisco, CA, November 1995.
- [9] N. Manohar and A. Prakash. The Session Capture and Replay Paradigm for Asynchronous Collaboration. In *Proc. of ECSCW'95*, pages 161–177, Stockholm, Sweden, September 1995.
- [10] L. Rowe and B. Smith. A Continuous Media Player. In *Proc. of the 3rd Int'l Workshop on NOSDAV*, pages 376–386, La Jolla, CA, USA, November 1992.