DECAF: A DIGITAL CLASSROOM APPLICATION FRAMEWORK

Allan Knight
University of California, Santa Barbara
Santa Barbara, CA 93106
email: aknight@cs.ucsb.edu

Kevin Almeroth
University of California, Santa Barbara
Santa Barbara, CA 93106
email: almeroth@cs.ucsb.edu

ABSTRACT
This paper outlines the design and evaluation of the Digital Classroom Application Framework (DeCAF). DeCAF is designed to provide an environment upon which multimedia applications for the classroom can be developed and evaluated. With the inclusion of multimedia in the modern digital classroom comes new paradigms for teaching. Also, the combination of the Internet and multimedia allow digital classrooms to accommodate many forms of distance learning. However, state of the art applications for digital classrooms are usually monolithic in nature and lack any integration with other similar applications. DeCAF provides an environment in which to develop and integrate such applications by treating a digital classroom as an event stream. The event stream abstraction is the key to better integration of applications and better means to evaluating new classroom paradigms.

KEY WORDS
Information Systems and the Internet, Digital Classroom, Automatic Authoring, Event Stream

1 Introduction

In recent years there has been a great deal of research and development in the area of multimedia. Also, with pervasive computing, the number of possible platforms is increasing as well. Everything from PDA's to Laptops to personal music players are able to handle some form of multimedia. Classrooms too, can be thought of as a platform, and they certainly are no exception to increasing amounts of research and the pervasiveness of multimedia. The digital classroom is quickly spreading to many campuses and is increasing in visibility.

With the inclusion of multimedia in the modern digital classroom comes new paradigms for teaching. With the help of the Internet and multimedia, digital classrooms are able to accommodate many forms of distance learning. *Synchronous learning*, learning that is done at a specific time for all participants, is one example of distance learning that has benefited from these maturing technologies. By allowing the streaming of multiple medias over the Internet, it is now possible for classrooms to be geographically distant yet allow for interaction and collaboration between multiple sites. *Asynchronous learning*, learning that is not necessarily done at the same time, is also possible. Students who otherwise could not attend a lecture are able to view lectures and other material at a time that is more convenient.

The first step in integrating multimedia into a traditional classroom setting is to improve the technology. For example, rather than presenting material on a blackboard with chalk, lecturers can present their material using presentation slides using PowerPoint or similar applications. These presentations seek to enrich the teaching experience by using a richer set of media in order to appeal to a broader base of students. As technologies have advanced, lecturers are able to incorporate video and audio into there presentations, allowing even more information to be conveyed to students. Finally, using the Internet, it has become possible to share material with more people over varying times and in different places. However, traditional teaching paradigms have not evolved as fast as these new technologies. Current research is looking toward developing new paradigms that are not only as good as traditional teaching aids (e.g. blackboards and chalk), but are better.

Currently many new paradigms for teaching in the digital age are being developed along with applications that help to realize these new paradigms. However, there are many problems with this growing collection of applications. First, there is no common framework on which to develop them. Most applications are monolithic and are not meant to be integrated with others to share what is happening in the classroom. Furthermore, there seems to be a lack of a feedback loop to collect information from participants. Finally, most systems tend to only focus on the semantic information of a lecture and pay no attention to other events or other sources of on-line and off-line information. Without addressing these issues it is hard to evaluate, other than subjectively, the effectiveness of new applications as well as new paradigms. A new framework is needed that addresses these issues.

The new framework should allow for the direct comparison of classroom applications and paradigms. With the current state of research it is possible to compare applications, but integrating them is nearly impossible, or at the very least time consuming. New classroom paradigms, however, are in a much worse state. In order to directly compare new paradigms they must be set up in a classroom with the proper capabilities. Each must then be installed run and then torn down so that the next paradigm can be installed. There is clearly a lack and a need for a common

framework to install these new paradigms and to allow for better integration of applications.

In this paper we propose a framework that addresses these issues. Our work is the Digital Classroom Application Framework (DeCAF) which consists of two components. The first component of DeCAF is an abstraction that treats the digital classroom as an *event stream*. All applications written in the framework are either an event producer, event consumer or both. The second component then deals with how an event consumer discovers what events are being produced within the classroom. Here we propose a publish/subscribe architecture. Producers publish to a registry what events they produce; consumers then query the registry as to which events are available; and finally, consumers subscribe to events that are of interest. All of these components are abstractions and do not require a specific medium on which to run. For example, the central authority may be a flat file or a client/server application on a network. Either way the abstraction provides the services needed for both consumers and producers.

For our framework to allow for the integration of applications and the direct comparison of new classroom paradigms it must be extensible and portable. Extensible means that each component must be flexible enough to accommodate as many new applications and paradigms as possible. As for portable, we mean the same for both components. Specifically we mean that it should be portable across multiple languages, media formats, computer platforms and execution paradigms. By being both extensible and portable, the framework seeks to limit as little as is possible the types of applications and paradigms that could be imagined in the future.

The goal of this paper is to present our design and evaluate it to determine if it can better provide for the integration of applications and evaluate paradigms for the digital classroom. In describing its design we also present simple examples to clearly communicate the purpose of each component in the design. However, evaluation is a more difficult task. Without quantitative metrics and other systems to compare, it is difficult to show that DeCAF fulfills its goals. By using a case study that examines how a new teaching paradigm can be created on top of the framework, we believe we demonstrate that DeCAF is both useful and capable.

The remainder of this paper is organized as follows. Section 2 summarizes current research related to DeCAF. Section 3 describes DeCAF in more detail. Section 4 evaluates the framework. The paper is concluded in Section 5.

2 Related Work

The related work we looked at can be divided into two groups. The first group is a collection of applications for digital classrooms or presentation environments in general that highlight the need for a common framework. The second group consists of research already done with regard to frameworks.

2.1 Digital Classroom Applications

While there are numerous papers about multimedia applications and entire conferences dedicated to research in this area, for example ED-MEDIA, we present a few examples here for the sake of brevity. Work at Xerox PARC and Bellcore looked to integrate text, audio and video into automated presentation tools. The work at PARC concentrates on a confederation of tools[7] used for the automatic collection of information from multimedia. PARC's tools lack any real integration of the tools and leave it until post-production work to manually integrate the media.

Bellcore's STREAMS[5] is directly targeted toward training and lecturing. The main idea is that the viewer knows best what is important, and therefore is allowed to choose whichever stream they deem important. All other streams are thumbnailed and can be selected at anytime. While leaving the choice to the user is novel, STREAMS is merely another application among thousands targeted to the classrooms of the future. STREAMS also overlooks a very important piece of information. It does not record which selections the users make. We believe this information is very important because it can help in evaluating an application as well as the presentation itself and should not be overlooked

The Classroom 2000 project[1] involves creating and producing digital multimedia that is useful to students. They use what they call "invasive technologies" to input the information. One example of an invasive technology is a digital whiteboard. Classroom 2000 is effective in using invasive technologies but does not do a good job of evaluating how invasive they are or allow for a full spectrum of less invasive technologies to be used. Also, they have again created a collection of applications that are not easily integrated with others as we are seeking to do.

2.2 Digital Classroom Frameworks

Seminal[8] concentrates on adding metadata to media streams and is specifically for automatic extraction of metadata based on cues from the lecturer. Predefined commands are used in order to more accurately extract important events such as the changing of topics or presentation slides. It is similar to our work in that it is a framework in which the media experience is enhanced by the addition of more information in the media stream. However, it is application-specific and does not look to create media events and therefore does not allow for easy integration or evaluation.

Similarly work, done by Brassil and Schulzrinne[3] also adds metadata to media streams, however in the form of synchronization events and content information. The services provided here are similar to those found on satellite TV and digital cable. Information about the title of

a program and its contents are added to the media stream along with timing information. This timing information is mostly useful for automatic synchronization between two different media streams. For example, advertisements can be inserted into a program stream by automatically inserting timing information into the stream. A streaming server for the advertisements then waits for the "cue" and starts the advertisements at the appropriate time. While this system is quite useful, its applications are limited and does not provide a framework that is flexible enough to handle a more diverse set of events.

Finally, Solar[4] is the research that most resembles our own. The authors in this work also use an event stream abstraction and the publish/subscribe paradigm for matching producers with consumers specifically for ubiquitous computing. However, they also add the abstraction of acyclic directed graphs. In reality there is no more complexity in the framework, they merely use the graph as an analogy to better explain the way their design works. All similar systems, ours included, work this way because of the nature of the design. Event producers in these designs act as the sources and event consumers the sinks. However, their work is directly applicable to ubiquitous computing environments and requires more complexity to handle the volatile and heterogeneous natures of such environments. We claim that our framework is just as flexible, perhaps more so, because their framework could be easily overlayed on top of ours.

3 DeCAF

As discussed earlier the state-of-the-art in digital classrooms is usually monolithic in nature and lacks any integration with other applications. Also, as new applications are identified, they usually have to be completely rewritten with little or no consideration of past or future work. This lack of integration and common abstraction increases development time and leads to digital classrooms that are merely an aggregation of independent applications rather than an entire framework. The major goal of DeCAF is to overcome this weakness and provide researchers with a common abstraction and framework in which to write future digital classroom applications and create new models for the classroom.

The goals of DeCAF are simply to provide an extensible and portable environment in which to write applications.It should be extensible so that any new events can easily be added and used by new applications. Also, DeCAF must be portable. We do not wish to limit the environment in any way. All devices, from high end workstations to devices with limited resources, must be able to operate within DeCAF. Finally, applications written within DeCAF should be able to run on something as small as a single machine, as well as in a highly sophisticated distributed environment.

In the following sections we describe the design of DeCAF and how the design fulfills the goals described previously. First we describe the main abstraction, the event stream, how it is aggregated, and how it is created. Then we discuss how the event stream is used and list some of the applications that can be built on top of this framework.

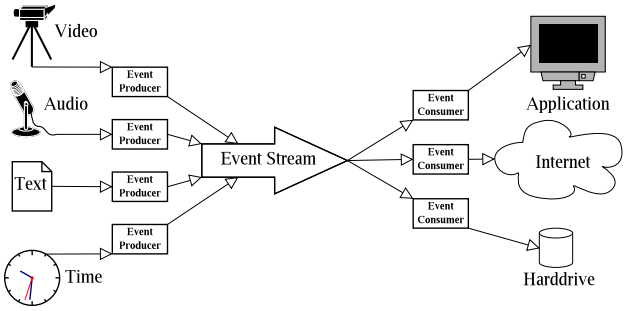

Figure 1. The flow of events in DeCAF

3.1 Event Stream Abstraction

To better understand the DeCAF design, it first helps to understand the main abstraction used within the framework, the event stream. First, we give a brief description of what the event stream is and then we discuss the individual events. Upon completion of this section it should be clear to the reader how the abstraction fits into the overall design of the framework.

3.1.1 Event Stream

The event stream is an abstraction that consists of all the events that may occur within a digital classroom aggregated together to form a stream. The event stream is merely an abstraction of a collection of data, which in this case, contains data pertaining to events within the classroom. The data itself may reside on network sockets or in a binary file or, better yet, an XML file. In order for the event stream to be extensible and portable, the abstraction must not dictate where and how the data is stored, only what is stored. The only specification needed is what data is important for each event type.

Figure 1 shows a graphical representation of the event stream abstraction. From this figure we see that the source of all events is a collection of event producers that direct events toward event consumers. How these events are produced and consumed are described in later sections.

The stream can be an aggregation of multiple sources such as video, audio or text. Event sources may reside on a single computer, or be distributed over multiple computers. In the case of multiple computers, they can be on the same network or on different networks connected via the Internet. Any list of sources is by no means complete. DeCAF must always allow for new events to be introduced to the stream or it is not extensible. It is the event stream extraction that is at the heart of the framework and that allows this framework to remain extensible and portable.

3.1.2 Events

At the heart of the event stream are the events themselves. They are responsible for carrying the information that allow applications to extract information from the media that make up a presentation. Events are also meant to be self organizing, allowing event consumers to effectively search for events that are pertinent to a particular task.

While it is impossible to anticipate all the various forms that events may take, it is possible to discuss what information all events have in common and how they are organized. Every event has at least two pieces of information: its partent identifier and its own identifier.

Events are basically organized hierarchically with object-oriented-style inheritance. At the root of the hierarchy is a single event called simply "Event". This is analogous to the class Object in Java, which is the base class for every class within Java. This design allows for great flexibility in finding events that consumers are interested in consuming. Consumers can find all events, all video events, specific kinds of video events, all the way down the hierarchy to specific video stream. Also very important is that this design also allows for events to be easily designed using functional, object oriented and other programming paradigms. The design is not meant to limit, in anyway, the way in which applications are implemented.

3.2 Event Stream Aggregation

The event stream itself consists of many types of events. Everything from audio, video, powerpoint, whiteboard material, and text can trigger events within the framework. Each of these event sources can then be aggregated together to create one complete event stream. While the entire stream may not actually exist in any one place, it helps to understand the abstraction if all the stream sources are aggregated together as one. From this stream, event consumers can be written to chose only those events they want to create further events that are fed back to the stream, or used directly within an application. To better understand how event streams can be aggregated let us look at different sources of events, and then see how they can be combined to create the entire event stream.

Let us first consider the events that may occur from a simple audio/video presentation within a classroom. Both the audio and video are fed into an event producer from which events are triggered. These events are then fed into the stream to create the entire event stream. Figure 1 shows this situation graphically to make clear how the event stream consists of all events that come from all sources and then flow to consumers.

3.3 Event Creation

What drives the event stream is obviously the event. The events are created by media processors. For each type of media there may be one or more processors that analyze the media and produce events that may later be consumed by other applications. For example, if we look at Figure 1, we can see the flow of information from raw media to events that are sent throughout the framework. As the raw media information flows from the left into the event processor, the media is analyzed and events are created. These events are then fed into the event stream for later use by consumers. To better clarify the process of event creation and to further illustrate the flexibility of the framework, we provide a few specific examples of how events are created from raw media.

The simplest example is video. Referring to Figure 1 we see that video comes in from a camera and is sent to a media processor. Events are emitted from the processor for later consumption. Events in this case could be as simple as starting and stopping the video. Each time the video is started or stopped it triggers an event. Applications that wish to use the events can listen and then use them. For example, an application may want to display the video, or another may capture the start and stop events to record viewer behavior. Here we see that by treating the raw media as a stream it leads to an abstraction of the information that is useful in many applications.

Of particular interest to the content producers is the creation of events based on how the user views the material. Previous research[8][3][6][4] has overlooked this source of events. If it were possible to record the interaction of users with the media itself, it would offer invaluable feedback to the producers of the presentations. A key feature of DeCAF is the ability to capture user interaction and feed it back into the event stream for later use, on-line or off-line.

Other forms of data can also be used to create events. Text is good example. For text we have identified three main types of events: the beginning of a text stream, for each word, and the end of a text stream. These three events when combined give us an entire text document. The forms of text documents that are input to the stream can be anything from raw text files to Microsoft Word documents. However, one source seems obvious from the application to digital classroom environments: transcription. By analyzing the audio data of a stream it is possible to create word events by using current speech-to-text converters. Here we see how one event stream can be used to not only create more events for the stream, but also how another media can be created from the event stream as well.

From our design we are able to produce a framework that allows for the inclusion of events from many different media and create an abstraction that makes this process both consistent and useful. The examples above are a short list of the possible ways the event stream can be created. These examples show that the abstraction is flexible and open to whatever future media events may be deemed important for applications within a digital classroom.

3.4 Event Use

Now that the events have been produced they need to be consumed. Events can be consumed in many different ways, just as they are produced from many different sources. For example, as the events come to a consumer they can be sent to an application. Examples of applications have been described in previous sections and several specific examples will be given in Section 4. It is further possible that these events can be stored on disk. The storage of events allows for later consumption by new applications not yet created when the presentation was originally given or for data mining purposes to allow for extraction of more information at a later date. It is also possible to stream the presentation to other off-site digital classrooms connected through the Internet. While it may not be feasible to send all events to all sites, it may be possible to send a subset of the events. Finally it is also possible for consumers to create more events for the event stream as mentioned previously. Once again, this is not an exhaustive list, but a list that demonstrates the flexibility of the system and how simple it is to provide more information off-line after a presentation has been given.

3.5 Connecting Consumers to Producers

Connecting event producers to event consumers uses a publish/subscribe model that is similar to what happens in transactions created in business-to-business applications. A producer of a product registers with a middle man and informs the middle man of what product they have to sell and how much the product costs. Then as consumers looking for products come along, they too communicate with the middle man and tell them exactly what sort of product they are looking for. The middle man the looks in its list of know producers and connects the consumer with the producer that can best meet its needs. The producer and consumer then communicate with each other to complete the transaction. This paradigm works well for system.

To connect an event producer to an event consumer a middle man is needed. For DeCAF, we create an event registry whose sole purpose is to allow applications running within the framework to find events necessary for it to run. When an event producer first starts it publishes its events to the event registry. These advertisements allow event consumers to know exactly what types of events are currently being produced in the framework and where they are located. Next, when an event consumer first starts it also contacts the event registry and subscribes to the events it needs. Finally, as new events occur the producer streams them to the consumers.

Once again this design is merely an abstraction and is not meant to be tied to any particular medium of communication. This architecture for publish/subscribe can easily exit on a network as well as in a single application. Also, the way in which discovery is done is also deliberately omitted. It could be done using broadcast calls over the network, hard-coded into an application, or done using peer-to-peer networks. The goal of the entire framework is to be flexible and not dictate implementation, and instead guide the design and provide a framework in which to create that design. The architecture presented here fits well within that goal.

4 Evaluation

In order to evaluate DeCAF we offer the following case study. This case study illustrates how DeCAF is used to support new models for teaching in a digital classroom. We look at model for a digital classroom based on a joint collaboration with a local company.

4.1 Case Study

A company here in Santa Barbara, QAD Inc.[1], produces software for manufacturing processes to customers around the world. As part of the solution, they also provide training services. These training services requires them to have numerous experts give presentations about specific processes and how QAD provides solutions for them. To maximize the information in these presentations they record them and put together multimedia presentations for customers to view at their own site. The presentations require a significant amount of manual post-production time. Our goal has been to develop a system to help them automate some of the process as well as help them find ways to further exploit the information in these presentations.

When one of their experts gives a presentation both audio and video is recorded and then used in post-production to create a presentation for viewing later. These presentations are hosted on their web site and include metadata embedded in the streamed video that allows for the automatic paging of slide show presentations. Also, the presentations are transcribed and then translated into all the languages that their customers use. However, both the paging of the slide show and transcription are done manually. In addition, it requires at least two people to record the event.

While a whole host of applications are required to give QAD the amount of automation they want, three specific application have been identified that they are interested in using. The first is an automatic transcription system. This system must not only transcribe the presentation but also group the text in chunks roughly equivalent to paragraphs. The second application is automatic camera tracking of the presenter. This application is especially useful since it has the potential to reduce the number of people required from two to one. Finally, they are interested in knowing what users do when they view the material offline. To better understand how well their experts are presenting the material, it is useful for them to know what material is viewed, or skipped, or viewed repeatedly. Currently they

[1] http://www.qad.com/

are only able to determine which presentations are viewed and for how long. It is our belief that DeCAF can be used to provide these applications.

Automatically transcribing presentations in DeCAF is very similar to what was described in Case Study 1. In this application the audio is first converted into text and then events are created to represent each of the sentences. These sentences are then used by an event consumer and grouped together producing an event for each chunk of text that is determined to be related. Later these chunk events can be used by an application that plays back the presentation and synchronize them with the play back of the video. Here again we see that if another application is needed to process the text, there would be no need to re-process the text.

More complicated than the transcription of the presentation is automatic camera tracking of a presenter. Because this problem has been addressed before[2], we do not have to re-invent the technology. A possible future project could see what information could be extracted from a presentation when the movements of the presenter are saved as events. To save these movement events, there first must exist events that indicate the start and end of recording. These events let the event consumers know when video is available and identify when the presenter moves. Once movement is recognized, an event is triggered and consumers can act accordingly. In this case it would consume the event and give commands to electronically controlled cameras to pan to the correct position to continue following the presenter. What is interesting about these movement events is that they can also be used later, offline. For example, if the producers of the video wish to see if there is a correlation between the amount of movement and the quality of the presentation, they can write an application that processes the movement events and compare it to other presenters and/or feedback provided by viewers of the presentations.

The final application, recording viewer feedback, is an example where an application is not only an event consumer, but also an event producer. To get viewer feedback an event consumer is created that listens for the beginning of the video, audio, text, and synchronization event streams. The application then begins streaming these media to the viewer. However, before starting the streams it also registers itself with the central authority as an event producer. These events include when the viewer presses play, stop, rewind, fast forward, or any other action that the viewer takes in the course of watching the presentation. These events act as an automatic feedback loop and are added to the event stream for the entire presentation. The information produced by this feedback loop can be used to help presentation producers make better presentations in the future by identifying segments that are perhaps boring or not well done, and also identify the segments that are the most watched. The ability to provide a viewer feedback loop is invaluable in creating better presentations in the future and even making those that are already completed even better.

These three applications, as well as others that will be developed in the future, show the flexibility and value of the framework. By treating the presentation as an event stream it adds value to the presentation, and provides the means by which to add further value well after the presentation is complete.

5 Conclusions

In this paper we have presented DeCAF, the Digital Classroom Application Framework. This framework represents an important step in better integration and evaluation of both applications and models for digital classrooms. DeCAF was designed to be both extensible and portable without putting too many restrictions on developers. Also using the case studies we were able to show the capabilities of the framework and how new models can be built on top of it. In the future we will be able to better create new applications, integrate them and from these new applications create new models for the digital classroom that provide a better educational experience than traditional classrooms.

6 Acknowledgments

The authors of this paper wish to thank Gregory Banks, Alex Cyriac, Ethan Smith and Phil Wittrock for their diligent search for related work.

References

[1] G. D. Abowd, C. G. Atkeson, A. Feinstein, C. Hmelo, R. Kooper, S. Long, N. Sawhney, and M. Tani. Teaching and learning as multimedia authoring: the classroom 2000 project. In *Proceedings of the fourth ACM international conference on Multimedia*, pages 187–198. ACM Press, 1996.

[2] M. Bianchi. Autoauditorium: a fully automatic, multi-camera system to televise auditorium presentations. In *Joint DARPA/NIST Smart Spaces Workshop*, July 1998.

[3] J. Brassil and H. Schulzrinne. Structuring internet media streams with cueing protocols. *IEEE/ACM Transactions on Networking*, 10(4):466–476, 2002.

[4] G. Chen and D. Kotz. Context aggrregation and dissemination in ubiquitous computing systems. In *Proceedings Fourth IEEE Workshop on Mobile Computing Systems and Applications*, pages 105–114, 2002.

[5] G. Cruz and R. Hill. Capturing and playing multimedia events with streams. In *Proceedings of the second ACM international conference on Multimedia*, pages 193–200. ACM Press, 1994.

[6] S. Duarte, J. L. Martins, H. J. Domingos, and N. Preguiça. A case study on event dissemination in an active overlay network environment. In *Proceedings of the 2nd international workshop on Distributed event-based systems*, pages 1–8. ACM Press, 2003.

[7] S. Minneman, S. Harrison, B. Janssen, G. Kurtenbach, T. Moran, I. Smith, and B. van Melle. A confederation of tools for capturing and accessing collaborative activity. In *Proceedings of the third ACM international conference on Multimedia*, pages 523–534. ACM Press, 1995.

[8] S. Rollins and K. C. Almeroth. Seminal: Additive semnatic content for multimedia streams. In *Proceedings of the Internet and Multimedia Systems and Applications (IMSA 2002)*, pages 174–179, August 2002.