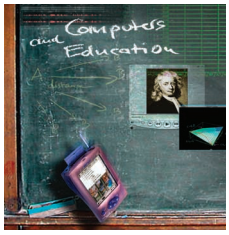
Architecting Multimedia Environments for Teaching



Thus far, developers have created only partial solutions for using computational equipment in education. Research must focus more effort on developing architectures capable of combining technologies that target the classroom and that allow specifying "what" rather than "how" tasks should be done.

Gerald Friedland
Karl Pauls
Freie Universität Berlin

Although surrounded by today's many technological enhancements, teachers remain utterly alone in front of their classes. Even in 2005, most teachers still rely on well-established primitive aids. For example, the chalkboard—one of history's earliest teaching tools—remains the preferred exposition medium in many scientific disciplines. Since the advent of computational devices in education, researchers have sought the means for properly integrating them and taking advantage of their capabilities.

The difficult task of architecting multimedia environments for teaching must start with a needs analysis. The most challenging task involves warranting reliability on the one hand, while accommodating opportunities for innovation on the other.

Thus, we propose building a reliable, ubiquitous, adaptable, and easy-to-use technology-integrating black box. Placing this system atop a service-oriented component model implemented on a platform-independent layer such as a virtual machine will provide the adaptability developers need. Loosely coupled components will accommodate a nonmonolithic approach and ease reuse. By reusing and enhancing components, the system will become increasingly reliable, while a building-block architecture will keep it manageable.

WHAT TEACHERS HAVE

The demand for computational equipment to use in education is surging. Several partial solutions already exist, but no one so far has put forth a global vision for using this technology. Nor have researchers devoted much effort to developing architectures that can combine technologies focused on the classroom with easily used designs.

To date, three e-learning approaches predominate:

- intensive use of slide show presentations;
- video recording lectures transmitted via fiber optics and, more recently, Internet broadcasting; and
- the creation of e-learning modules such as dynamic Web pages, flash animations, or Java applets.

Slide show presentations enable good visualization and smooth lecture performance. The instructor plans the presentation's structure up front, taking into account all required resources. Visual elements such as tables, diagrams, or images can be directly presented to the audience. Further, computer-generated slides can be printed out so that students don't need to copy the content for later review.

However, slide show presentations often appear static because everything must be planned in advance, leaving few possibilities for the teacher to adapt the content in interaction with the students.

Usually, slides present content in note form, structured as bullet-point lists, which dramatically restricts the lecturer's freedom of expression. Often, the instructor must deliver information out-of-band

0018-9162/05/$20.00 © 2005 IEEE

Slide shows, video recordings, and e-learning modules result from the search for answers to the educational field's most pressing concerns.

and unencapsulated in the slides because drawing diagrams by hand is easier and more intuitive and doing so does not require hours of preparation.

Students sometimes feel overwhelmed when a huge number of slides are displayed in rapid succession. Especially in math and physics, the journey is the reward, so students need to focus not on the results presented in the slides but on the development of the thoughts that led to them.

Recording a video of the entire lecture—including a picture of the board and the lecturer, along with an audio track—lets students follow the lecture remotely and recall previous sessions. To record or transmit classes, standard Internet video broadcasting systems have become popular. This allows taking advantage of the availability and straightforward handling of state-of-the-art video broadcasting software.

Existing solutions either focus on recording and transmitting a session or using videoconferencing tools to establish a bidirectional connection, or feedback channel. This approach does not support the teaching process as such, and it also introduces additional issues. Technical staff must be on hand at least for setup and maintenance. The more professional approaches require camera work and audio recording personnel. Standard video Webcast tools make the recording's technical quality inappropriate for educational content. Text and drawings, either from slides or the chalkboard, tend to be poorly encoded either because video compression omits sharp edges or because delivering the content smoothly requires a high-bandwidth Internet connection, and some students cannot easily access the video from a remote location.

Educational mini-applications such as dynamic Web pages, flash animations, or Java applets can be used for presentation as well as for individual training by the student at home without imposing restrictions on the content or representation. However, using conventional authoring systems, the ratio of production time to the duration of the produced learning unit tends to be wildly disproportionate, mainly because traditional teaching know-how does not easily match contemporary authoring tools.[1]

In addition to technical efforts, using these units to structure didactical content for the Web requires a huge amount of work. Given the lack of a standard, the implementation and interfacing possibilities vary as much as their purposes. This makes reuse or adoption by third parties almost infeasible, while reassembly usually requires a complete rewrite.

WHAT TEACHERS NEED

It makes sense to assume that slide shows, video recordings, and e-learning modules directly result from the search for answers to the educational field's most pressing concerns. An analysis points directly to the following needs:

- support for classroom teaching that includes the possibility of integrating educational mini-applications;
- tools that support the preparation of classroom teaching; and
- synchronous and asynchronous remote-teaching support.

Clearly, given its status as their core job, teachers must consider classroom instruction their first priority. This makes it likely they will refuse any tool that does not leverage the experience and practical knowledge they have gained in a lifetime of teaching. An excellent teacher should remain excellent whether aided by electronic devices or not. Further, becoming familiar with a given technology should require little time. Any tool must thus ensure that it conforms to the teacher's established working habits while adding value to the teaching experience. Given that every teacher has a different perception of what constitutes a good lecture, instructors must be able to change tools according to their preferences and ideas.

A smooth lecture performance depends on the quality of the preparation, which in turn consists of gathering content, structuring the lecture, and preparing didactic elements such as charts, figures, and pictures. Multimedia elements can be useful didactic tools in this context.

The increasing use of such elements requires even greater preparation effort. To avoid redundant work—such as presetting the lecture on paper—computer-based education tools must support this process conveniently. This involves preserving as much freedom as possible while allowing as much structure as needed. A teacher should retain control over the amount, order, and elements of the material to be included in the lecture. An experienced teacher, for example, can hold an excellent lecture spontaneously, backed by life experience only.

Synchronous remote teaching, such as videoconferencing, can provide courses that resource constraints would make impossible otherwise. In addition to helping students recall past lecture con-

tent, asynchronous remote teaching provides benefits that assist students in reviewing past lectures to catch up on missed content or prepare for examinations, and, if they are physically impaired, gives them greater overall access to lectures. Additionally, institutions promote lecture recording and broadcasting because they anticipate that these archives will enhance the institution's knowledge base and enhance its prestige.

Synchronous or asynchronous teaching offers a valuable enhancement for students, but teachers will implement it only if doing so entails as little overhead as possible. In essence, teachers need a tool that substantially assists in their preparation and delivery of lectures, allows easy integration of various didactic media, and optionally supports synchronous remote teaching with little or no lecturer overhead. This involves realizing the twin priorities of providing the highest amount of assistance on one hand while permitting the greatest degree of freedom during class on the other.

SURVEYING THE JUNGLE

Schools and universities form a sprawling, heterogeneous playground. Developers who want their software system to achieve sustained success must build it to survive in an environment that consists of a variety of software and hardware configurations. More importantly, they must make it adaptable to different software ideologies by, for example, avoiding design decisions based on their political biases toward various operating systems.

Baseline requirements

The software must fit the existing hardware infrastructure and should be easily adaptable to working with other multimedia applications. Developers should also avoid excessive concentration on the construction of proprietary specialized solutions, a trend that has resulted in the notable absence of generic, sustainable, and reusable approaches to e-learning. The hardware and software that support classroom teaching must eliminate as much overhead as possible.

Teachers who enhance their lessons with computing devices should still be able to step into the classroom and start lecturing as usual. Both teachers and students must take the technology for granted—which cannot occur until it becomes ubiquitous. This, however, requires the seamless integration of structural changes or functional enhancements—such as the addition of new media, changes in technical formats, or simply an upgrade to new hardware.

Realistic expectations

Vendor advertising claims to the contrary, lecture halls or seminar rooms are not professional recording studios. Consequently, most current systems will not yield professional-quality recordings just by plugging a microphone into a sound card and starting the lecture. A realistic approach should also emphasize reliability: Systems must continue working after their individual parts fail, at least at the level of providing switchover or backup functionality.

Developers who want their software system to achieve sustained success must build it to survive in an environment that consists of a variety of configurations.

Successful remote teaching requires an awareness of the targeted students' technical prerequisites and takes into account future technological or target-group shifts. For example, following a remote lecture for the first time presents a formidable psychological barrier when the participant must first install the client software. We can't assume that all students have an Internet connection, let alone a high-bandwidth connection. For example, a 2003 survey of engineering students in Berlin revealed that although 93 percent had Internet connectivity, more than 50 percent had only a modem connection.[1]

Given these statistics, we advise broadcasting at different quality levels, splitting the content into different streams, and providing the remote viewer with the choice of turning off individual audio, video, or slide streams for a given broadcast. Additionally, content should be distributable by offline means such as DVD.

Seeking sustainability

The more specialized a solution, the lower the probability of its reuse—which directly affects its sustainability. Further, just as in software engineering, monolithic approaches hinder partial reuse. For example, extracting individual slides from a presentation recorded entirely as compressed video can be cumbersome. Additionally, the content of many course topics, curricula, and presentations undergoes rapid changes because of technological innovations, legislative alterations, or cultural developments.

Therefore, development should focus on creating content rather than administering it. Exclusively building content-management systems or performing excessive research on metadata will not help to achieve this goal nor reduce costs.

A proposed software environment for multimedia-enhanced teaching should offer more than just the management of individual educational units. It must provide a complete solution that allows the

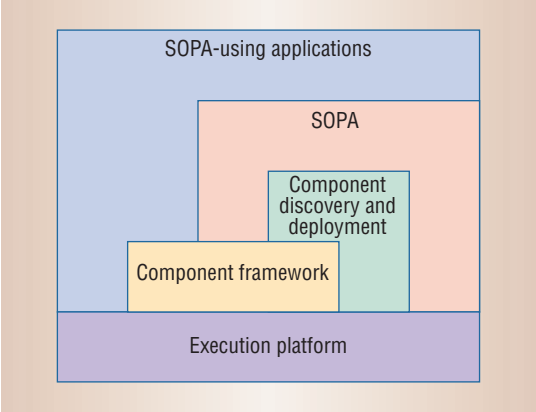


Figure 1. Self-Organizing Processing and Streaming Architecture (SOPA). This framework manages multimedia processing and streaming components organized in a flow graph that features autonomous assembly of stream-processing components.

integration of educational mini-applications and fundamentally supports the creation and distribution of generated content.

Integrating technologies for teaching in the targeted environments while supporting reusability and cooperation between different organizations and systems requires high flexibility on the one hand and high reliability on the other. Ideally, solutions will integrate seamlessly into a teacher's workflow using the hardware the institution provides, while available personnel can easily manage configuration.

INSIDE THE BLACK BOX

What does this reliable, ubiquitous, adaptable, and easy-to-use technology-integrating black box look like from the inside?

The component-based software engineering approaches proposed over the past several years for application areas include the tools built with the Eclipse Rich Client Platform (www.eclipse.org). We propose that these approaches offer a perfect solution for creating classroom-supporting applications. Further, service orientation can provide loose coupling between components inside a specific framework,[2,3] which allows for their dynamic download and deployment from remote sources. These components subsequently provide their services in the local framework and add functionality to an application during runtime, as needed.

Generic classroom installation

In a generic scenario, after installing the classroom software, users configure it according to the application's specific needs. To achieve this, they use tools that specify what will be used rather than how it will be implemented. Subsequently, the system analyzes its environment, downloads components from remote repositories as needed, and assembles these services into a composition that provides the required functionality.

An audio recording wizard provides one example of such a configuration and environmental analysis tool.[4] An expert system presented via a GUI wizard guides the user through the systematic setup and test of the sound equipment. The result, an initial modified composition description, contains a set of filtering services needed for the pre- and postprocessing of a given recording. While the lecture is being recorded, the system monitors and controls important parts of the sound hardware. For example, it detects and reports a range of handling errors and hardware failures. The system also simulates and automatically operates standard recording-studio equipment such as graphic equalizers, noise gates, and compressors.

Remote teaching

Another example, a whiteboard application, captures participants' written text, then stores and transmits this data to interested clients over the Internet. To provide additional functionality, the teacher uses a wizard to choose from a list of services—provided by locally or remotely available components—that can be seamlessly integrated into the lecture. When the lecturer chooses a certain service, such as a bubble-sort visualization mini-application, the actual component downloads and the whiteboard core displays the service. The same service can be used on the client side, possibly provided by a different component.

In a video-streaming scenario, various services could be used to convert the content into different formats. A receptor service receives the incoming connection requests, then chooses and configures the right converter services to mediate between the captured video type and the format type that can display it on the client's software.

The availability of downloadable components provides one of our proposed approach's most beneficial elements. Yet the question of how to build these remote repositories remains unanswered. Given the generic approach used to build components that provide their functionality via services, one task is to define service contracts in the form of interface descriptions. After this, arbitrary parties can share their component implementations in those repositories. The information specified in a service contract must be decided in the context of the specific domain. Given this restriction, syntactical interface descriptions combined with a few metadata properties will suffice in most cases.

The approach we propose may seem abstract, but it can be implemented pragmatically using established technologies.

SOPA

As Figure 1 illustrates, the Self-Organizing Processing and Streaming Architecture (SOPA) (www.

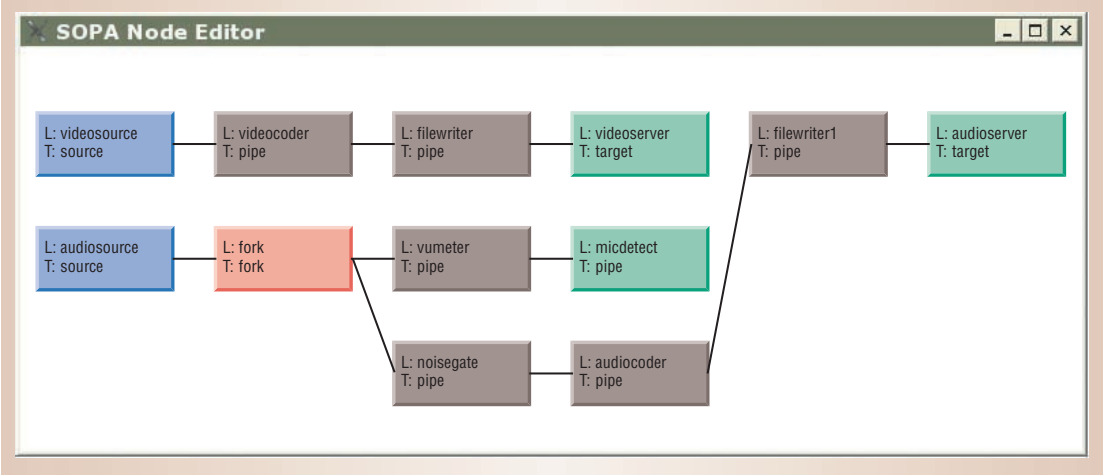


Figure 2. A streaming graph automatically assembled by SOPA to transmit and filter an audio and video broadcast.

sopa.inf.fu-berlin.de) works well with our approach. This framework manages multimedia processing and streaming components organized in a flow graph that features autonomous assembly of stream-processing components.[5] The dynamically organized processing graphs use components from various distributed sources. SOPA installs these components on the fly according to a graph's requirements, which derive from its specific purpose and are assembled according to an application's needs. An XML file describes the graph that glues these components together.

SOPA uses general properties to describe nodes—for example, to tell the system to select an audio codec that compresses to a certain bandwidth. The graph's structure can be changed and its nodes updated while the system runs, as Figure 2 shows. The implementation uses Oscar (http://oscar.objectWeb.org), Richard S. Hall's open source implementation of the OSGi framework,[6] as the underlying component model.

SOPA currently focuses on multimedia-processing and Internet-streaming applications on either the server or client side. On the server side—on a video streaming server, for example—SOPA integrates and manages codecs according to the connecting client's capabilities at runtime. Further, it supports seamless and transparent reconfigurations and updates to the client by dynamically adapting stream processing to user demands.

For each specific demand, SOPA uses Eureka—a Rendezvous-based component discovery and deployment engine[7]—to assemble a special processing graph that uses components discovered locally or in remote repositories. When accessing remote repositories, SOPA retrieves the requisite components from the remote source and deploys them locally.

SOPA ultimately seeks to ease the development of applications that need an extensible streaming and processing layer while also decreasing the administrative maintenance workload. More specifically, SOPA provides a round-up solution that serves as an extensible framework for managing multimedia components. SOPA synchronizes different independent multimedia streams, such as slides and video streams, and uses an application-independent approach to describe the handling of concrete content such as converting from one multimedia format to another.

E-CHALK

The E-Chalk software system (www.e-chalk.de) captures and transmits chalkboard-based lectures over the Internet. Conceived and supervised by Raúl Rojas and developed by his group at the Freie Universität Berlin,[8-10] E-Chalk enhances classroom teaching by integrating the multimedia features of modern presentation software with the traditional chalkboard. The software simulates a chalkboard using a touch-sensitive computer screen on which the lecturer works using a pen. This approach preserves the didactical properties of the traditional chalkboard while helping instructors create modern multimedia-based lectures.

Inside E-Chalk, SOPA handles on-the-fly streaming and recording of lectures. E-Chalk comes with a set of nodes that can handle and convert different media types and the necessary XML graph description. When a teacher starts E-Chalk to record or transmit a lecture, the media graph has already been built and the system is already live and online. During media graph construction, the system also checks for updates. At this point, the system updates any nodes that require it.

Using SOPA, students also could receive E-Chalk streams using QuickTime or Windows Media Player. To accomplish this, a receptor—a generic media node—waits for an incoming connection and checks its type. Depending on the connecting client's type, it restructures the given media graph, performs searches, and downloads nodes that support the new media types, using Eureka if necessary.

The E-Chalk software works with a variety of hardware components that instructors can substitute for the traditional chalkboard. For example, a lecturer can write on a digitizer tablet or on a

Figure 3. E-Chalk system used in a lecture hall at Technical University Berlin.

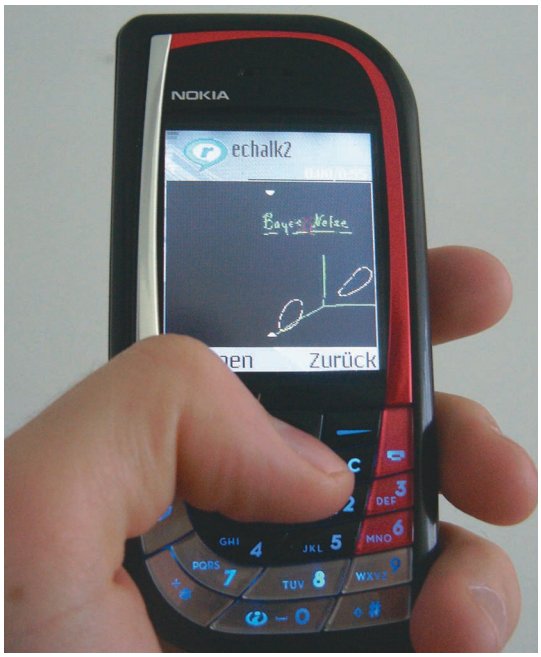


Figure 4. An E-Chalk lecture replayed on a mobile device, with audio and board content replayed at 16 Kbps.

tablet PC, using an LCD projector to display the computer screen's content against a wall. Instructors also can use digitizing whiteboards, like those shown in Figure 3, while an LCD projector displays the screen content on a suitable surface.

The software transforms the screen into a black surface upon which the instructor can draw using different colors and pen thicknesses. Scrolling up and down on the board vertically provides an unlimited writing surface. The lecturer can use an eraser to delete part or all of the board's content. Images from the Web or a local hard disk drive can be placed on the board during a lecture. E-Chalk provides access to CGI scripts to help implement the interfacing of Web services.

The instructor also can use educational mini-applications in the form of Java applets pulled from the Internet on the board. When the lecturer uses a reserved handwriting-recognition color to draw strokes on the board, the handwritten input passes to a mathematical formula recognizer. The recognizer transforms the input and passes it to other components such as the Mathematica or Maple interface. This can be useful for presenting partial results or to annotate the plot of a certain mathematical function. When the lecturer needs a computation's result, Mathematica or Maple answers with text or an image.

E-Chalk provides another means for interacting with third-party components: *Chalklets*. These applications interact only through *strokes*—they recognize drawings and gestures from the screen and respond by drawing their results on the board.

When an E-Chalk lecture closes, the system automatically generates a PDF transcription of the board content. The transcription can be generated either in color or in black and white.

Macros—E-Chalk's means for preparing lectures in advance—provide a prerecorded series of events that an instructor can call up and replay on the board during a lecture. To do this, the instructor draws the portions of the lecture to be stored in advance. During the lecture, the macros replay either at their original speed or at an accelerated rate determined by the user. Automatically generated macros can be used for visualization.

E-Chalk records all events from the screen or tablet, together with the lecturer's voice and an optional video. The lecture can also be transmitted live over the Internet and synchronized with video-conferencing systems, such as Polycom ViaVideo, for student feedback.

Remote users connect to the E-Chalk server to view everything as seen in the classroom. They can choose to receive the audio and, optionally, a small video of the teacher. A complete E-Chalk lecture with dynamic blackboard image, audio, and video can be maintained with a connection speed of roughly 128 Kbps. Without the video stream, the required connection bandwidth drops to at most 64 Kbps.

Java-based playback provides the most convenient means for following a lecture. In this case, the viewer requires nothing more than a Java-enabled Web browser. It isn't necessary to install a plug-in or client software manually.

Other options include following the lecture in MPEG-2 format on a DVD, a Java-enabled PDA, or a third-generation mobile phone that runs RealPlayer, as Figure 4 shows. When viewing archived lectures, the remote user sees a control console like the one shown in Figure 5 and uses typical VCR tools such as pausing, fast-forwarding, and rewinding to regulate the content flow.

EXYMEN

The Exymen software framework (www.exymen.org), shown in Figure 6, seeks to fill the role of a universal cross-platform multimedia editor[11] by becoming the rapid prototyping tool of choice for developers of new media formats and codecs. Exymen's developers also promote the tool by offering it as a free download for editing media content.

The editor provides a cross-platform GUI and defines generic data structures and operations for handling time-based media abstractly. Developers can plug in components that fill the abstract data structures with content by providing concrete format-specific handlers. These extensions can be loaded and updated without workflow interruption at runtime. Components can use both the framework and other components, assisting extension development through software reuse.

Developers can provide components to the system from remote locations using Eureka. Several plug-in components have already been developed for Exymen, while all types of E-Chalk content can be edited individually or synchronously.

Exymen can edit audio content supported by the Java Media Framework—such as wav files, QuickTime, or MP3 formats. It also can edit and synchronize Web-exported PowerPoint slides. Exymen can use SOPA's media nodes for editing archived lectures because it shares the local component cache with SOPA. A dedicated Exymen component reads in media graph descriptions to find conversion paths.

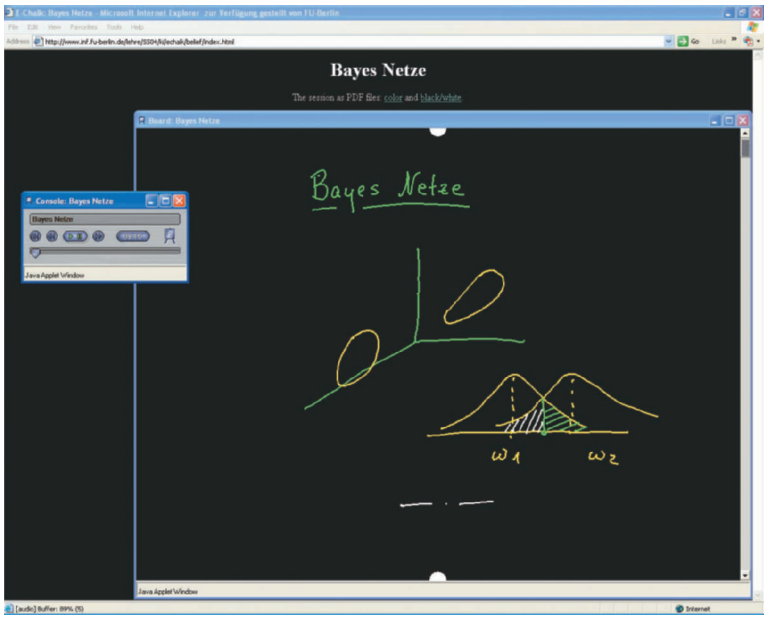


Figure 5. A lecture recorded with the E-Chalk system and replayed in a Java-enabled Web browser.

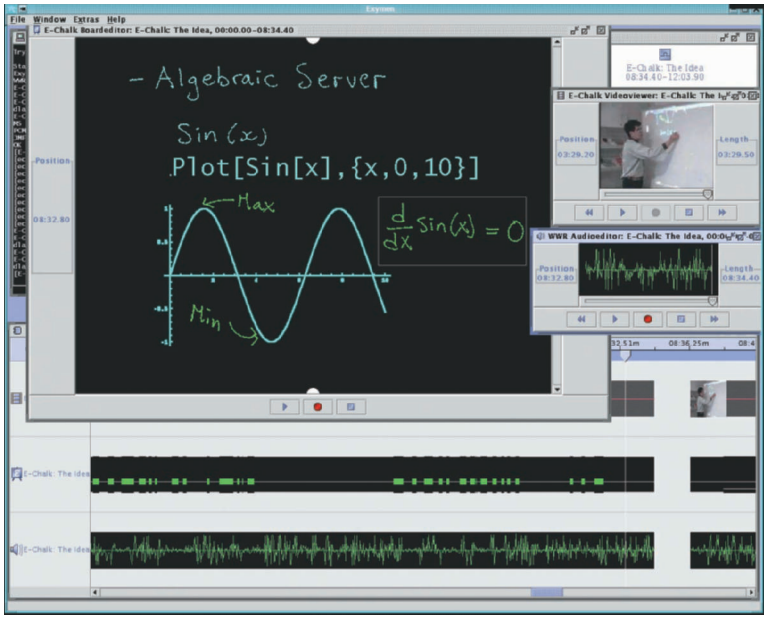


Figure 6. Exymen software framework, used here to edit a recorded E-Chalk lecture.

Many open questions remain concerning our approach to architecting multimedia environments for teaching, but we believe that by using our requirements analysis to classify proposed aids and by building solutions that adhere to our proposals, many mistakes and much futile or redundant work can be avoided. ■

Acknowledgments

The E-Chalk system, an ongoing project at Freie Universität Berlin since 2001, was conceived and is supervised by Raúl Rojas. Several others have contributed to the system, including Kristian Jantz, Ernesto Tapia, Christian Zick, Wolf-Ulrich Raffel, Mary-Ann Brennan, Margarita Esponda, and—most noticeably—Lars Knipping in his doctoral dissertation.

References

1. G. Friedland et al., "E-Chalk: A Lecture Recording System Using the Chalkboard Metaphor," *Int'l J. Interactive Technology and Smart Education*, vol. 1, no. 1, 2004, pp. 9-20.

2. R.S. Hall and H. Cervantes, "Challenges in Building Service-Oriented Applications for OSGi," *IEEE Comm. Magazine*, May 2004, pp. 144-149.
3. H. Cervantes and R.S. Hall, "Autonomous Adaptation to Dynamic Availability Using a Service-Oriented Component Model," *Proc. Int'l Conf. Software Eng.* (ICSE 2004), IEEE Press, May 2004, pp. 614-623.
4. G. Friedland et al., "The Virtual Technician: An Automatic Software Enhancer for Audio Recording in Lecture Halls," *Proc. 9th Int'l Conf. Knowledge-Based & Intelligent Information & Eng. Systems*, LNCS, Springer, to appear Sept. 2005.
5. G. Friedland and K. Pauls, "Towards a Demand-Driven, Autonomous Processing and Streaming Architecture," *Proc. 12th Int'l IEEE Conf. Eng. Computer-Based Systems* (ECBS 2005), IEEE Press, 2005, pp. 473-480.
6. R.S. Hall and H. Cervantes, "An OSGi Implementation and Experience Report," *Proc. IEEE Consumer Comm. & Networking Conf.* (CCNC 2004), IEEE Press, 2004, pp. 394-399.
7. K. Pauls and R.S. Hall, "Eureka—A Resource Discovery Service for Component Deployment," *Proc. 2nd Int'l Working Conf. Component Deployment* (CD 2004), LNCS 3083, Springer, 2004, pp. 159-174.
8. G. Friedland, L. Knipping, and R. Rojas, "E-Chalk Technical Description," tech. report B-02-11, Dept. Computer Science, Freie Universität Berlin, 2002.
9. R. Rojas et al., "Teaching with an Intelligent Electronic Chalkboard," *Proc. ACM Multimedia 2004, Workshop on Effective Telepresence*, ACM Press, 2004, pp. 16-23.
10. L. Knipping, "An Electronic Chalkboard for Classroom and Distance Teaching," doctoral dissertation, Dept. Computer Science, Freie Universität Berlin, 2005.
11. G. Friedland, "Towards a Generic Cross-Platform Media Editor: An Editing Tool for E-Chalk," *Proc. Informatiktage, Gesellschaft für Informatik*, Konradin Verlagsgruppe, 2002, pp. 230-234.

Gerald Friedland *is a PhD candidate and researcher at the Center for Digital Media in the Department of Computer Science, Freie Universität Berlin. His research interests include intelligent multimedia technology for electronic learning. Friedland received an MS (Dipl.-Inform.) in computer science from Freie Universität Berlin. He is a member of the DIN-NI 36 committee that cooperates with ISO SC-36 in creating e-learning standards. Contact him at fland@inf.fu-berlin.de.*

Karl Pauls *is a PhD candidate and researcher in Klaus-Peter Löhr's Software Engineering and System Software Group, Department of Computer Science, Freie Universität Berlin. His research interests include security and access control in distributed systems, component-based software engineering, model-driven development, and distributed systems. Occasionally, he collaborates with Gerald Friedland to work on SOPA's underpinnings. Pauls received an MS (Dipl.-Inform.) in computer science from Freie Universität Berlin. Contact him at pauls@inf.fu-berlin.de.*