The Weiner Lecture Archives : An Ontology-Driven Interface for Viewing Synchronized Lectures and Notes

Gene Zhang
Sean Carr
Sameer Iyengar
Hava Edelstein
Albert Liu
Dan Garcia

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-135
http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-135.html

November 8, 2007

Acknowledgement

We wish to acknowledge the generous support of the Larry Weiner Trust for funding this project. Thanks to Lecturers Mike Clancy and Brian Harvey for useful feedback throughout, to the helpful folks at ETS for hosting our video streams, and to project alumni Cassandra Guy, Babak Pahlavan, Matthew Ng, Jun Kitagawa, Aaron Steele and Steven Chan for their work developing early versions of WLA. Finally, thanks to Larry Rowe for his leadership and foresight with the BIBS and CMT projects.

The Weiner Lecture Archives : An Ontology-Driven Interface for Viewing Synchronized Lectures and Notes

Gene Zhang
University of California, Berkeley
zyc@berkeley.edu

Sean Carr
University of California, Berkeley
seancarr@berkeley.edu

Sameer Iyengar
University of California, Berkeley
sameer@berkeley.edu

Hava Edelstein
University of California, Berkeley
hava@berkeley.edu

Albert Liu
University of California, Berkeley
albert_j_liu@berkeley.edu

Daniel D. Garcia
University of California, Berkeley
ddgarcia@cs.berkeley.edu

ABSTRACT
For several years, the lectures in our introductory Electrical Engineering and Computer Science (EECS) courses have been videotaped and webcast, mainly as an aid to students with time conflicts that prevent them from attending class. We present the Weiner Lecture Archives -- a project to identify, archive, filter, and make available the best of these lectures with their notes on the web. We provide a hierarchical, ontology-driven interface to entire courses, which allows users to choose any topic and/or subtopic to view, from a small snippet of one lecture to one that spans many lectures. Once the topic is chosen, our system launches RealPlayer to play the lecture video in one window while showing synchronized lecture notes or slides in another window. By the spring of 2007, we had finished encoding our department's entire four-course introductory sequence into this system. Student use and feedback has been encouraging, and we hope to expand to other EECS courses in the near future.

Categories and Subject Descriptors
K.3.1 [**Computer Uses in Education**]: Distance Learning

General Terms
Documentation, Design, Human Factors

Keywords
Distance learning, webcast, Synchronized Multimedia Integration Language, SMIL, Slide synchronization, topic hierarchy, ontology

1. BACKGROUND
In the mid-1990s, colleague Larry Rowe and his students in the Berkeley Multimedia Research Center were designing two systems that would serve as the architectural foundations for this project. First, they were building the hardware and software infrastructure of the Berkeley Internet Broadcasting System (BIBS) [10], which provided live remote viewing and on-demand replay of course lectures through streaming audio and video over the web. We were one of the earliest to offer free university lecture webcasts, and it pushed the limits of bandwidth and processor capabilities of the day.

Figure 1. Larry Weiner

In parallel, Rowe and his students were working on the Berkeley Continuous Media Toolkit (CMT), which provided synchronized continuous media playback [7]. The media could be anything: video, audio, text, and even commands to control the Tcl-based client (say, to load a particular lecture slide in a window at a certain moment). BIBS became so successful that the University's Educational Technology Service (ETS) [4] took it over and grew its capabilities and capacity[1]. The computer science four-course introductory sequence was webcast and archived every semester. This was a great resource to students who missed class or wanted to review a confusing lecture, and even to autodidacts from

[1] ETS currently offers free webcasting and archived playback, in RealMedia streaming format [9], for 47 of the biggest lecture courses on campus, as well as hundreds of special events by guest speakers and luminaries.

around the globe who wanted to learn computing on their own. However, the webcast system had several shortcomings.

First, only the *raw* lecture webcasts were available, replete with deprecated semester-specific information (e.g., announcements for talks or midterms that occurred years ago) and wasted lecture time (e.g., fumbling with a USB drive, mid-lecture breaks, etc.). Second, slides / notes for these lectures were not linked to the webcast, so viewers always had to go hunting through outdated class websites to find them. Once the notes were downloaded, there was no way of synchronizing them with the video in RealPlayer [9], so viewers had to manually scroll them in a separate window while watching the lecture. There was also no connection to our anonymous course surveys [1,6], so semesters that were highly rated were indistinguishable from those that were duds. Finally, the webcasts were presented by date recorded instead of by curriculum topic, so content that spanned lectures required extra downloading, and there was no easy way to quickly jump to a short snippet within a lecture.

In June of 2002, UC Berkeley alumnus Larry Weiner (Figure 1) passed away, but left a trust fund to our department for educational resources. We decided to honor his name by building a system, the Weiner Lecture Archives (WLA) [11], to augment the existing free webcast infrastructure but address all the problems we'd noted.

We began by consulting the course surveys to find the most recent, highest-rated semester for each of the intro courses. We then gathered and posted all the notes for that semester, and hired a former A+ student for each course to go through all the lectures to encode the appropriate time-code slide transitions, allowing for synchronized playback of slides and video. During this process, the student filters out all but the purest curriculum content (by not recording the time-codes for the offending segments or by recording them under a special category such as "administrative" or "obfuscated"), essentially *cleansing* our webcast. This process can be done in real time, so this step requires approximately 45 hours for a 15-week semester-long course. Finally, we wrap our content with the gentle, navigable interface of an ontology tree based on all of the course topics, labeling each node by topic as well as playback length.

In the following sections, we describe our interface and provide a detailed description of the implementation of all the varied components that make up the project. We'll have a discussion of the usage we've noted, feedback we've received, reflection on a critical architectural decision, and conclude with a discussion of where the project will go in the future.

2. INTERFACE

From the WLA home page, a student selects one of the four lower-division computer science classes we currently offer from the big tabs at the top and is taken to a dedicated class page. From there, they see all of the main topics covered in that course for that semester. Exploration of the tree reveals that each main topic branches off into several sub-topics. Figure 2 shows this for the CS3 course ontology. Once the user chooses the topic (or sub-topic, or sub-sub-topic, etc.) they wish, they click the link.

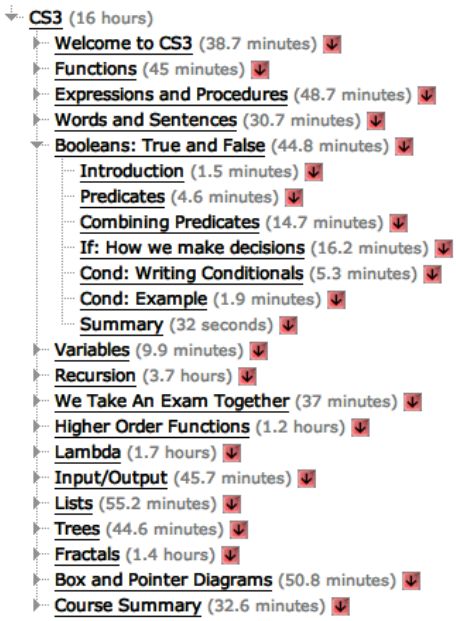

Figure 2. Screen capture of the CS3 course ontology, after the user has chosen to expand the "Booleans: True and False" subtopic. The clip length for each topic is shown in parenthesis. The down-arrow at the right of each topic displays only that sub-tree.

This brings up RealPlayer with the video cued exactly to the requested clip in one window and the synchronized notes in the right. As the video proceeds, the notes auto-advance. Figure 3 shows what results if the user clicks the "Introduction" subtopic under the "Booleans: True and False" topic for CS3 and watches for 19 seconds. Note that the overall displayed video length is exactly the length of the clip requested, with 0:00 labeling the start and 1:31 (1.5 minutes, as advertised) the end. We abstract away all information about the originating source video files (e.g., the actual time within the webcast(s) that the clip was extracted, from which webcast(s) they were culled, etc). If the clip happens to contain lectures from multiple webcasts recorded on different days, the user notices nothing when the video seamlessly crosses a day boundary, aside from the fact that the instructor instantaneously changes clothes (hopefully)! The user need only consider course content, and leave the heavy lifting (of determining the sources required to *deliver* that content) to us.

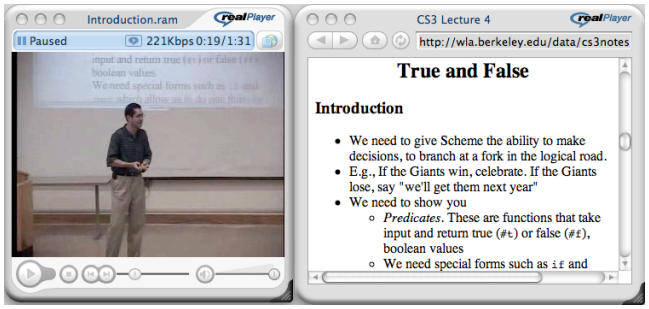

Figure 3. The lecture webcast video synchronized with HTML notes. As the lecture progresses, the notes auto-scroll to the current topic via web anchors. The video can be scrubbed, and the video and notes will re-sync.

2.1 Integration into UC-WISE

UC-WISE is an interactive collaborative learning environment [2] currently used in some of our intro CS courses. In a UC-WISE course, students have a very lab-centric experience: they spend one hour in lecture per week instead of the traditional three, and *six* hours per week in lab instead of just two. Lab activities range from taking online quizzes, reading web pages, engaging in collaborative "brainstorms", responding to directed questions in a communal newsgroup setting, and interacting with an interpreter.

Now, thanks to WLA, that can include watching a short lecture video clip! This can also address the loss of two hours per week with the esteemed "sage on the stage" (who may be a gifted, inspirational speaker). The curriculum developer can easily locate the relevant clips by navigating through our ontology just as a normal user would. However, instead of *clicking* on the link, they copy the link *location* and paste it into the UC-WISE authoring environment. They can choose to make the video compulsory or optional, depending on the context. There are some subtleties still to be worked out (e.g., we now need to make headphones required to watch the clips in lab, how do we let them know when content has been updated, etc.), but this integration with UC-WISE holds great promise.

3. IMPLEMENTATION

3.1 Slide Synchronization

WLA uses the Synchronized Multimedia Integration Language (SMIL) [12] for most of its synchronization between lecture slides/notes and streaming videos. SMIL lets us declare sequences of multiple video files (useful for viewing lecture topics that span multiple days of lecture), and start & end times for playback for each video file. It also lets us specify various actions for RealPlayer at arbitrary points during playback. For example, we use this to synchronize all of our lecture slides, by instructing RealPlayer (via a SMIL extension developed by RealNetworks, Inc) to display a different URL for a different lecture slide at specific times during video playback.

All notes are first located from the course web page or individual instructor and copied to our web server. WLA currently supports HTML and raw text notes without any pre-processing. In our system by way of SMIL, time codes are linked to different URLs, so scrolling is made possible through URLs that link to page anchors peppered throughout the document. PDF files require a pre-processing script step that splits each one into individual pages (which themselves are assigned a unique URL). PowerPoint slides require must be first converted to a 1-up PDF file, and processed accordingly. Most slide transitions can be preserved if the PDF creator chooses the "save *with* animations". The most common technique for putting PowerPoint slides on the web involves rasterizing each slide into images, a far inferior solution to PDFs that preserve the object properties of the presentation.

3.2 Ontology System Design

In this section we will describe the internals of our system and what factors and decisions led us to the current design. Starting with the low-level data structures, we will build up to the algorithms that power the core of the user interface.

3.2.1 Data Structures

The main data structure in our system is a tree of topics. We chose MySQL because we were familiar with it, it was already set up on our servers, and it would likely be on any servers we might migrate to in the future. The three main database entities we have are *videos*, which reference external video files and contain meta-data about them; *video clips*, which define clips within a video using a specific start and end time and link the clips with their places in the lecture notes; and *tree nodes*, which define the relationships between nodes that create the ontology structure, store a topic name for each node, and reference any video clips associated with the node.

3.2.2 Database Abstraction Layer

Given the complexity of the relational tree structure we built and the redundancy and synchronization requirements it has, we decided to build an abstraction layer so the application code does not have to deal with managing it. The idea is similar to the *Model-View-Controller* paradigm, although we did not decouple our controller code from our view. Having a clear data abstraction layer with a defined interface saved us countless development hours as we re-factored the system several times. Initially, the abstraction layer consisted of low level *getters* and *setters* for all of our entities. As the application became more complex, however, we added higher and higher-level methods so that complicated data manipulation tasks still seemed simple to the application code. For example, moving an entire sub-tree from one place in the tree to another is as simple as making a call to: `moveSubtree(src_node_id, new_parent_node_id)`. Other than being easier to use, providing methods for complex tasks like this allows the abstraction layer to be responsible for managing transactions rather than relying on the application code to do it, which eliminates the problems of nested transactions.

3.2.3 Optimizations

3.2.3.1 AJAX On-Demand Loading

The first iteration of our ontology-driven interface loaded an entire topic tree (or sub-tree) before displaying it and therefore was extremely slow for a tree of any reasonable size. Once the tree was loaded, expanding and collapsing sub-trees was relatively quick, although it required looping over all descendants (of the clicked node) and hiding or showing them. The current design eliminates the need to always load the entire tree. Using AJAX, we delay the loading of sub-trees until they are needed, only load one level of the sub-tree at a time, and only load more when it is requested. In addition, expanding/collapsing a sub-tree is faster because we now only hide/show the immediate children of the sub-tree. Using AJAX on-demand loading was a big performance win and changed the feel of the site from sluggish to responsive.

3.2.3.2 Lineage Field

To efficiently select all the clips for a given sub-tree we created a "lineage" field that lists all the node's descendents. Having this field allows us to select, with a single database query, all nodes with the same ancestry. Therefore, when the user clicks on a topic to watch the video we can select all nodes that are descendants of that topic node in a single query. We also have another field that defines the absolute ordering of these nodes. We use to efficiently sort the results so the aggregated video clips are shown in the correct order. The key benefit here is we only need one query

whereas traversing the tree would have required many queries, had we only kept track of a node's parent and/or children.

3.3 Architecture

Due to certain constraints, our PHP web server, MySQL database server, and video-streaming server are all on separate machines and networks. The speed of a single MySQL query doesn't suffer much from this setup, but there remains a fair amount of overhead involved, which is one of the reasons we made all possible efforts in our application to reduce every task to a single query. We use a laptop on the same network as the web server as a backup database server. Nightly backups are made from our master server to the backup server, and also to our test server to ensure that the test data closely mimics our production data. The basic components of our architecture are fairly generic and are readily available in most web application environments without having to navigate around our server constraints. Thus, it would be fairly easy to replicate WLA in another setting, provided that a high-bandwidth RealVideo streaming server were available to power the actual video delivery, as ETS does for our project.

3.4 Data Entry and Ontology Layout

3.4.1 Topic Tree Delineation

To facilitate the data entry process, we created a standard text file format which allowed the editors to specify all necessary video, clip, and node information, as described in 3.2.1. After uploading the timing files, the topic tree is fine-tuned if needed using a web-based topic tree editor.

3.4.2 Ontology Development

The process of developing a final configuration of nodes in the topic tree from the raw videos and lecture notes involved several steps and decisions, each with its own considerations. We chose start and end times of each clip so that each corresponded to a single topic, ideally lasting around three to four minutes. Keywords were used in the naming of nodes to aid searches; for instance, all nodes corresponding to examples in lecture began with "Example". By the nature of the topic tree, nodes representing broader topics occupied shallower positions in the tree. Where possible, each of the siblings in a group represented roughly equally important/broad topics, and each non-leaf node branched off into five to seven children in order to limit the depth of the tree without confusing users with too many choices at each node [8]. Siblings were arranged so that topics that were more basic, broad, important, and/or earlier in the course appeared first.

3.4.3 Data Entry Personnel, Resources, & Strategies

One student was assigned to each course to be presented on the website. These students had each achieved a grade of A+ in the course that they were assigned. To facilitate the viewing of the webcasts, instead of streaming the webcasts from the server, each student was given a local copy of all the webcasts for their assigned course on DVD. Some strategies were used to speed up the production of timing files. For example, we discovered Enounce 2xAV [5], a plug-in for RealPlayer that allowed videos to be played back at speeds greater than real time. The accuracy of lecture start and end times were later spot-checked by jumping to those precise times in the raw video to confirm that the times were exactly the ones desired.

4. DISCUSSION

We went live with WLA in the spring of 2005. It featured an early, non-ontology-based interface design, and listed only CS3, our computer science service course. We advertised it heavily to the students in that semester's UC-WISE lab-centric CS3, and waited eagerly to see the adoption. The usage statistics we saw (which have since become unavailable) showed three spikes: one right after our announcement, one before the first midterm, and one before the final. Whether our viewers were star students wanting to hear another perspective on the material or slackers who had panicked and were looking for a savior, we don't know. Since we just recently went live with the remaining three introductory computer science courses (CS61A, CS61B and CS61C) in the spring of 2007, we don't yet have good data on their usage. What we do know is that even *without* all the benefits of WLA, we had gained an active global community of distance learners from just the *raw webcasts*! Here are a few of the many thanks we've received:

"Just a quick note to thank you and Berkeley University for putting Machine Structures CS61C on the web. ... Great stuff, thanks. I'm a 55 year old computer tech/engineer"
– Tony Atkinson, Norwich Norfolk UK

"I am a 35 year old electronics engineer. I have always loved programming and as a hobby. I just wanted to say...great stuff!"
– Naim Lesmer, DENMARK

"Thank you for making the web casts and much of the course material available online for free. I am learning a great deal from your lectures alone. Having the lecture slides available separately is a plus. I am an autodidact that greatly appreciates having learning resources, such as your lectures, freely available via the internet" – Gary Buczkowski, Tennessee, USA

"I am currently a student at the University of Houston and watch the Webcast to supplement my computer architecture class material. Once again thank you and the University of California Berkeley for making this service available."
– Carlos J. Restrepo, Texas, USA

4.1 Dueling Webcasts

In retrospect, we believe we made the right design choices along the way. At the dawn of WLA, we evaluated other systems for presenting synchronized slides and video. The most promising was one our colleagues across campus at the Digital Chemistry Project were developing called PRISM [3] that interfaced with Macromedia's Flash Communication Server in a very novel way [3]. Instructors first converted their PowerPoint slides into Flash format before class. Then, *during* class, instead of running a traditional PowerPoint slideshow, they displayed their slides using PRISM. Meanwhile, in the room at the back of the hall, the video from the camera was being encoded into Flash format onto the server in real time. The beauty of the system was that the slide transitions were also being recorded in real time (with appropriate time-codes) onto the server, so that the moment lecture concluded, the webcast (with synchronized notes) was ready!

When we saw a demo, however, the first thing we noticed was that the video "stuttered" and was unwatchable! That is, every second, both video and audio would freeze for a quarter second, and this was on a fast campus wired network connection with the fastest laptop available at the time. There were other deal-killers

as well – we'd have to ask every instructor to adopt the PRISM system, the slide converter program wasn't perfect, and we'd have convince the campus ETS to pay for a Flash media server to serve our data. We decided their system wasn't yet ready for prime time, and instead focused on building our system around the existing campus webcast framework.

5. FUTURE WORK

One issue we have yet to deal with is what to do with deprecated content, e.g., a Java 1.1 demo. Our current practice is to mark it as obsolete (making it unavailable to a user), and replace it with updates if available. Ideally, we would provide a natural way to navigate through our deprecated material as well ... heck, some of our viewers may still be running Java 1.1! We have yet to decide how to present this content in a way that is clean and consistent with our current design.

Another limitation of our system is that we do not support arbitrary meta-data tags for ontology nodes. This has great benefits for search, especially if combined with tags that users could supply. While we're letting users contribute content (Web 2.0 meet WLA. WLA, Web 2.0), why not release the keys to the castle and let them submit their own webcast annotations? Certainly the clips and accompanying ontology would have to go through an approval process, but opening up the indexing process could make it much easier to add other courses to our archive. A huge step in this direction would be to find or develop PowerPoint-embedded software that automatically recorded slide transition time-codes during lecture, as PRISM did. One would still need to filter an instructor's "back, back, back, (oops too far!), forward, (long pause for explanation), forward, forward" clicks into "jump-back-2-slides, (long pause), jump-forward-2-slides", but it might be possible to write scripts to help automate that process to search for common instructor patters.

We do intend to add more courses to our archive, probably starting with our Electrical Engineering introductory courses and then moving on to the EECS upper-division and maybe even graduate courses. We also hope to add one-time lectures, culled from our weekly colloquium series, research group talks and thesis presentations. In the long term, following in the footsteps of BIBS, we hope to eventually become adopted by the campus and become fully integrated in their webcast infrastructure.

6. CONCLUSION

We have presented the Weiner Lecture Archives, a system built by U. C. Berkeley undergraduates over the course of several years, that provides an ontology-driven interface to synchronized webcast video and slides of the "greatest hits" of our computer science introductory courses. Our hierarchical topic tree allows the viewer to easily navigate to the content they want, and with the click of a single link, watch anything from a 1-minute clip for a simple demo to the 1.5-day video of an entire course. This feature has been critical to the integration of our repository with the UC-WISE lab-centric instruction project, which will be infusing our synchronized video snippets into its curriculum. The feedback we have received from all over the globe has been unanimously positive, and is quite encouraging. In the near term, we hope to add more courses and one-time lectures to our archive, to provide an even richer experience for our worldwide community of distance learners.

7. ACKNOWLEDGMENTS

We wish to acknowledge the generous support of the Larry Weiner Trust for funding this project. Thanks to Lecturers Mike Clancy and Brian Harvey for useful feedback throughout, to the helpful folks at ETS for hosting our video streams, and to project alumni Cassandra Guy, Babak Pahlavan, Matthew Ng, Jun Kitagawa, Aaron Steele and Steven Chan for their work developing early versions of WLA. Finally, thanks to Larry Rowe for his leadership and foresight with the BIBS and CMT projects that got the webcasting and slide-synchronization balls rolling on campus.

8. REFERENCES

[1] Chen, Carl, GoodProfOrNot: About, http://goodprofornot.sourceforge.net/about.html

[2] Clancy, M., Titterton, N., Ryan, C., Slotta, J., and Linn, M. 2003. New roles for students, instructors, and computers in a lab-based introductory programming course. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education* (Reno, Navada, USA, February 19 - 23, 2003). SIGCSE '03. ACM Press, New York, NY, 132-136. DOI= http://doi.acm.org/10.1145/611892.611951

[3] Cuthbert, A., Kubinec, M., Tanis, D. O., Ieong, F., Wei, L., and Schlossberg, D. 2005. Advanced technology for streamlining the creation of ePortfolio resources and dynamically-indexing digital library assets: a case study from the digital chemistry project. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems* (Portland, OR, USA, April 02 - 07, 2005). CHI '05. ACM Press, New York, NY, 972-987. DOI= http://doi.acm.org/10.1145/1056808.1056812

[4] Educational Technology Services, Webcasting, http://ets.berkeley.edu/Webcast/

[5] Enounce, Enounce 2xAV, http://www.enounce.com/products.shtml

[6] Eta Kappa Nu, Course Surveys, http://hkn.berkeley.edu/student/CourseSurvey/

[7] Mayer-Patel, K., Simpson, D., Wu, D., and Rowe, L. A. 1996. Synchronized continuous media playback through the World Wide Web. In *Proceedings of the Fourth ACM international Conference on Multimedia* (Boston, Massachusetts, United States, November 18 - 22, 1996). MULTIMEDIA '96. ACM Press, New York, NY, 435-436. DOI= http://doi.acm.org/10.1145/244130.244458

[8] Miller, G. A. (1956). "The magical number seven, plus or minus two: Some limits on our capacity for processing information". *Psychological Review*, 63, 81-97

[9] RealNetworks, http://www.realplayer.com

[10] Rowe, L. A.; Harley, D; Pletcher, P; and Lawrence, S. "BIBS: A Lecture Webcasting System". *Berkeley Multimedia Research Center, TR 2001-160*, June 2001. http://bmrc.berkeley.edu/research/publications/2001/160/

[11] University of California, Berkeley, The Weiner Lecture Archives, http://wla.berkeley.edu/

[12] World Wide Web Consortium, Synchronized Multimedia, http://www.w3.org/AudioVideo/