Lectures On Demand: the architecture of a system for delivering traditional lectures over the Web.

Mauro Dolzani and Marco Ronchetti
Dipartimento di Informatica e Telecomunicazioni
Università di Trento – Via Sommarive 14, 38050 Povo (TN) - Italy
mauro.dolzani@studenti.unitn.it, marco.ronchetti@unitn.it

Abstract: The rapid expansion of the Internet has finally brought a wider availability of large bandwidth connections reaching the homes. This means that the content that can be delivered through the Internet is dramatically changing. We run an experiment of eight whole university courses: we web-cast the lectures, and also we made them asynchronously available both on-line and off-line (on CD's). We based our experiment two software systems: the first one, called e-Presence, was developed at the University of Toronto, Canada. The second, called LODE, was developed at the University of Trento: it was inspired by the experience gathered using e-Presence, and incorporates a series of requirements that aim at overcoming some limitations we encountered when using the first system. In this paper we describe requirements and architecture of the LODE System.

1. The background: scenario and our experience

The availability of wide band Internet connections is a very recent achievement: in 2001 a paper presenting an e-learning experience in Finland (the most advanced country in Europe, as far as technology access is concerned) quoted "Synchronous communication or even delivery of asynchronous video lectures was out of question because of the low bandwidths available at students' homes" [1]. Over the last few years things are changing at a dramatic pace: Point Topic's latest analysis of the world DSL market [2] shows that the total number of DSL lines in the world has doubled in less than 9 months. Overall, the cost of broadband fell by 22% worldwide over the past three years. In USA, as of April 2004, more than half (52%) of a key demographic group - college educated people age 35 and younger - has broadband connections at home. Broadband penetration is currently at 28% in South Korea. In Europe the largest penetration is in Belgium (19%). It's 13-15% in Germany, France, Italy and the United Kingdom and in some smaller European countries. Broadband adoption is occurring at high speed, according to Point Topic which finds a 55% year-over-year increase in the number of worldwide lines

In Italy we are also assisting to a mass adoption of DSL technology, thanks to a very aggressive advertisement campaign run by all the major telecommunication companies operating in the country. It is now not uncommon for an average family to have access to wide band connections. Moreover, even the effective bandwidth delivered on DSL lines has increased in a significant manner over the last 12 months.

It is therefore now becoming possible to deliver multimedia content on demand to a vast audience. This fact is having a dramatic impact on the way Internet is perceived and used: for the first time we assist to the sale of direct transmission of the videos of sporting events like soccer games over the Internet. It is probably the dawn of the TV on-demand. Such scenario will of course also have a strong impact on e-learning, and we can today experiment new paradigms that are made possible by the new technological achievements.

At the Università di Trento we tried to explore the possibilities offered by the new scenario. We experimented with the on-line delivery of lectures. We considered a whole spectrum of computer-mediated delivery of didactic events, going from real-time synchronous, to on-line asynchronous, to off line. Our goal was to evaluate costs and impact, to understand what the most effective way to exploit the new opportunities is, and to gather requirements for improving the application we used as a "first-order approximation".

We started by using a research product of the Knowledge Media Design Institute (KMDI) of the University of Toronto, Canada. Their system is called "ePresence". Their main goal was to make web-casting highly interactive, scalable and robust, engaging, and accessible both in real-time and later via structured, navigable, searchable archives [3]. The client runs in a normal web browser, enhanced by a plug-in chosen among Real Player, Apple's QuickTime and Windows Media Player. E-Presence is designed to support manly events like MS PowerPoint-based presentations, as reflected by its design. The idea is that the most important cognitive factors are the voice, carried by an audio stream, and the slides that illustrates the concepts that the speaker is talking about. The actual video is less important from the point of view of carrying information, although its presence may be very useful by showing

gesture, expressions, and contextual indications. Therefore the video carried by the system is small so as to save bandwidth, but very fluid and sufficient for carrying the needed information. The most significant portion of the screen is dedicated to an image of the PowerPoint slide, which can be enlarged in a pop-up window by double clicking on it. The system allows webcasting events with a small delay (approx. 30 seconds). When the speaker moves to a new slide, such slide is presented to the remote viewer. During webcast it is possible to interact with the speaker by posing questions through an integrated chat. A mediator person sitting in the room where the events takes place monitors the chat and delivers the questions to the speaker. The whole event (audio + video + slides) is saved on disk, and after some post-processing is archived in a form suitable for on-line asynchronous viewing. When accessing the events asynchronously, users see the video, the slides and also a list of the slides' titles and a timeline. Marks on the timeline show slide transitions and semantic marking of sections. Users can navigate the video by clicking on the slides' titles: the system synchronizes video and slide to reflect the user's choice. Jumps to different times can also be achieved by clicking on the time bar. A view of the system in asynchronous mode is presented in Fig. 1.

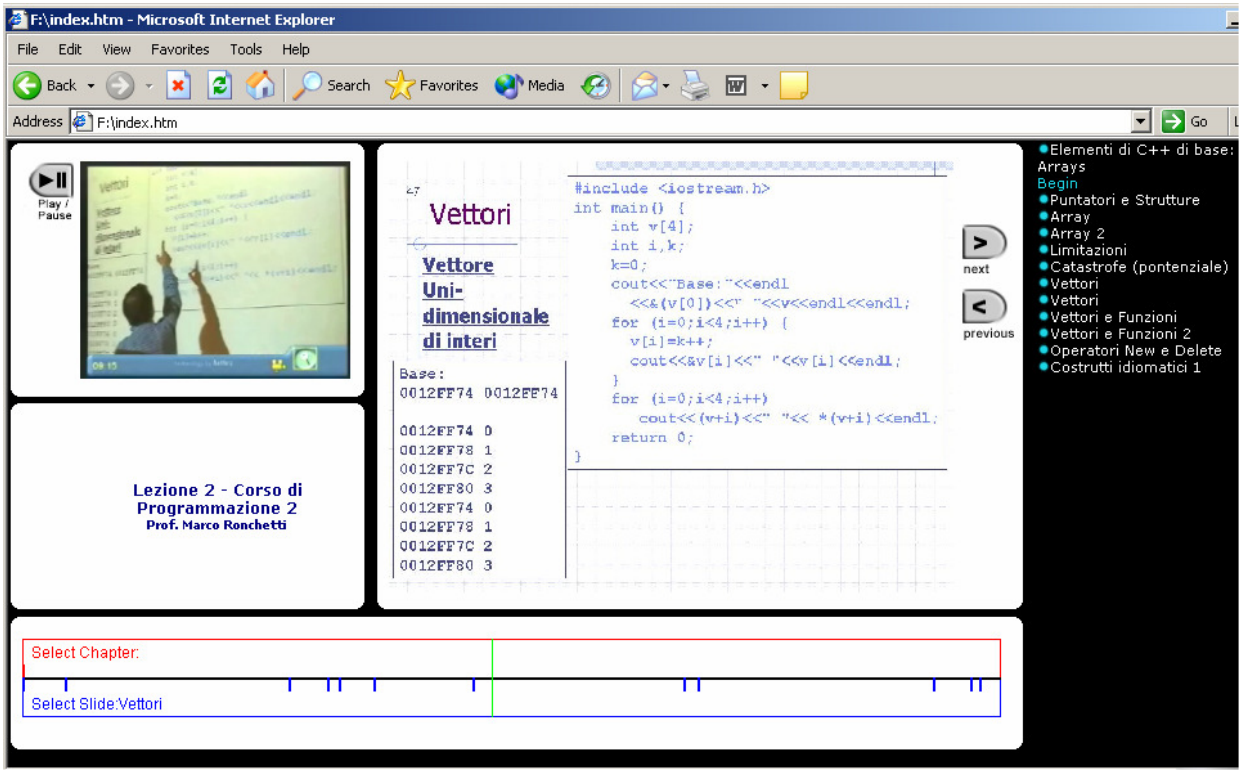

Figure 1. *A view of the ePresence client in asynchronous mode.*

Over three semesters, we used the system to record all the lectures of seven courses, for a total of approximately 350 hours. Students' feedback was excellent. Most of our students regularly follow the courses in class, but we also have a number of working students who, due to their job, almost never show up in class. Such students range between 5 and 10 percent. They typically try to catch up by studying on books, on teacher's notes and on other student's notes. The availability of asynchronous lectures is for them an extremely valuable resource, and they were simply enthusiastic about the service. The initiative has however advantages also for "regular" students. Among them we mention:

- the ability to recover lectures lost due to forced absence (e.g. illness);
- the possibility to chose a different temporal organization, deciding not to be present at some lecture (elective absence);
- support for foreign students who might have difficulties with the Italian language (they would benefit from the possibility of re-hearing portions of lectures);
- support for Italian students attending to courses given in English (some courses are);
- the possibility to review pieces of a lecture at any time, to check their understanding or their notes.

Early reports of our experience are available elsewhere [4,5].

2. Obtaining new requirements from our experience

Although we were quite happy with the e-Presence system, we found that one could imagine a system that was more suited to our needs, even by dropping some of the nice features of the Canadian software. For instance, although we can imagine situations in which synchronous webcasting would be precious, it was almost of no use in our case: if students can attend lectures at the right time, they prefer to be in the classroom. If they cannot come, being able to see the event in real time is not an important advantage, if they can anyhow attend to the event at any time starting from a few hours later.

We found instead necessary to support also those students who do not have a large bandwidth internet access in their homes. Trento lies in the hearth of the Alps, and some of the valleys nearby are not yet served by DSL. Also, DSL connections can be expensive for some families. Therefore to reach our goal we provided lecture collections on CD-ROMs. A full 50-hours course fits on 6 CDs, or on a single DVD.

We needed to extend the concept of the repository for asynchronous fruition to include also a zipped downloadable version of the files. A request in this sense came mostly by students living out of town, not having DSL at home but having friends who do: for them the most convenient alternative was to ask their friend to download the image of a CD using his/her fast lines, and burn the CD remotely avoiding the need to come to the University to get a physical copy.

As we mentioned, the video carried by the ePresence system is small so as to save bandwidth. PowerPoint based lectures this is more than enough. However in cases when the teacher moves to the blackboard, the small video screen becomes a limit. Also, it is impossible to use the system for lectures that are fully based on the use of blackboard. Having a larger size video was therefore among our wishes.

System transportability was an issue. The system we were using needed a server for delivering asynchronously the material, and an acquisition station that was composed by two PC's, both of them having an analog acquisition card and a stream splitter: the video coming from an analog video camera was delivered though the splitter to both machines: one was devoted to webcasting, the other to storing the video stream for later processing. The whole system ended up being rather bulky and costly. We needed a more agile solution, to allow to set up the system in less than 5 minutes between one lecture and the other. A lighter solution, based on a less noticeable infrastructure, would also have minimize the (already small) impact on the lecture itself.

Another point was to make the system really easy to use for the operator who records the event. We want to keep as low as possible the running costs of the system. Therefore we cannot require the presence of a technician, and we would like to choose the operator among the students who are anyway present. The training time must be minimal, the system should be as intuitive as possible. Also, the post-processing (i.e. the process that brings to the publication of the acquired lecture) must be fully automated.

We needed to make it possible to integrate the system as a part of the Learning Management System, delegating to the LMS the access control. The lectures are an important asset of the University, and as such they should be protected. Different people have different feelings about this issue. Some teachers just do not care, and the are happy to make their lectures freely available to anyone in the world, like many people do with learning material. Others desire that only their students have the possibility of seeing the lectures. The system should allow for any desired protection level, and the best option is to rely on the LMS for granting the correct rights to the users.

We needed more control on the production of the videos. For instance, some lectures have one or more frontal sections, and other sections in which the teacher assigns some exercise to be done in class. While in this mode the teacher goes around the class discussing with single students or with small groups, and helping them. For our goals it does not make any sense to capture the exercise phase, so it should be possible to suspend and resume video acquisition whenever needed. The software we were using, being principally concerned with webcasting, did not foresee such possibility.

Editing the lectures must be possible. In some cases the teacher might want to delete a portion of the lecture. Also, when re-using lectures over the next year some changes might be needed. When teaching, it often happens to have topics that span over more than one lecture. Alternatively, a lecture might be composed by more fragments dealing with different topics. It would be nice to recompose the material according to semantics rather than leaving it organized like the temporal constraints dictated.

Another interesting feature is the possibility of attaching additional documents at any given time. Such feature could have several uses. For instance the teacher could add additional notes, like clarification of some point, errata, indications for further study etc. Also, it would be possible to allow students to attach annotations for personal use, of for sharing them with friends, with all other students and/or with the teacher.

3. Architecture of the system

As a first choice we decided to focus on asynchronous delivery, leaving aside for the moment the simultaneous video acquisition and webcasting, since this last feature is not really necessary for our goals (though we designed the new system leaving an open possibility to add that feature later, as a complement to the system).
An important decision regarded the target hardware for the video acquisition. As we mentioned, it was for us very important to choose a low-cost hardware setting that could be easily movable. In this sense, a laptop would be ideal. Moreover, most laptops today come with an IEEE 1394 (also known as FireWire or i.Link) port. With such port, interconnecting a digital camcorder becomes matter of a cable (as opposed to attaching analog camcorders to PC's, which requires an expensive video acquisition card). We found that Pentium-4 based laptops with enough memory (512 MB at minimum, better if 768 MB) and a clock frequency of at least 2 Gigahertz would offer enough power to deal with real-time acquisition. A device for accessing wireless network can be useful. We also found that Centrino-based laptop under Linux seem to currently have some problems in delivering the expected performance for this kind of applications. The hardware choice also meets our budget requirements: the whole hardware setting, composed by a laptop, a digital camera, a radio-microphone and the needed cables costs now approximately 2000 Euros.
Another important technical decision regarded the choice of the video stream type. We decided to use MPEG 4, as it offers the possibility of coding video at a good resolution while maintaining a relatively low bandwidth occupation. MPEG 4 is supported by Apple QuickTime, which became our favorite plug-in to be embedded in the client's browser. QuickTime is available for Apple Mac OS-X and for MS Windows. At present a porting on Linux is not available: however in the Linux arena other MPEG-4 players are available, and the same is true for Unix, so that our system can have clients on all major OS platforms.
Before proceeding to the description of the system architecture, it is useful to recall that what the final user sees is a browser window that is composed by several parts, as shown in figure 2.

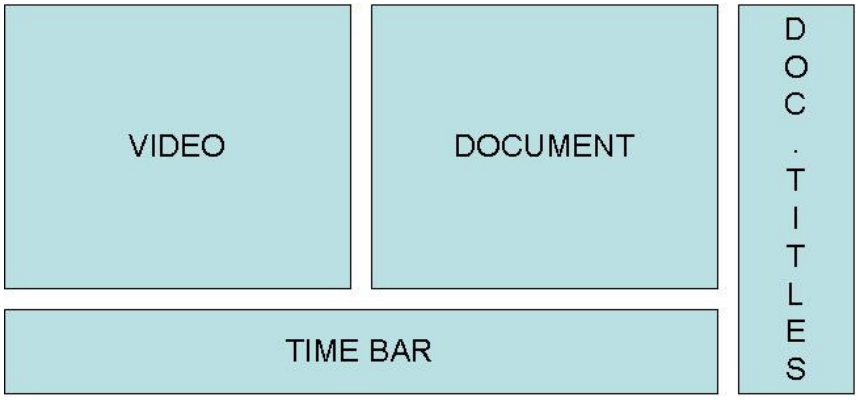

Figure 2. *The main components on the client's window.*

The components are:

- the Video area where the video stream is shown;
- the Document area where the reference document is presented;
- the Document Titles area where the user can find the list of all documents presented during the video, and by using them can quickly jump to the section where the chosen one starts being discussed;
- a time bar area where a progress bar runs giving a feeling of the elapsed time, and allows the user to jump to different times with a click.

The layout is inherited from the ePresence project. In our case however the dimension of the Video and Document areas can change on the flight on user's demand. The user can chose among different video resolutions: small (144x180 pixels) medium (288x384 pixels) or large (520x562 pixels). To make the enlargement of the video possible, the document area is correspondingly reduced; in any case by double clicking on the document area the user can have an enlarged version presented in a pop-up window. In any case the user can at any time switch to a different configuration with just one click on a suitable button. Figure 3 shows three images of the screen captured with the different options. As one can see, the blackboard becomes easily readable when the largest video area is

chosen. At present we are also evaluating a different layout, that is based on independent windows rather than on HTML frames.

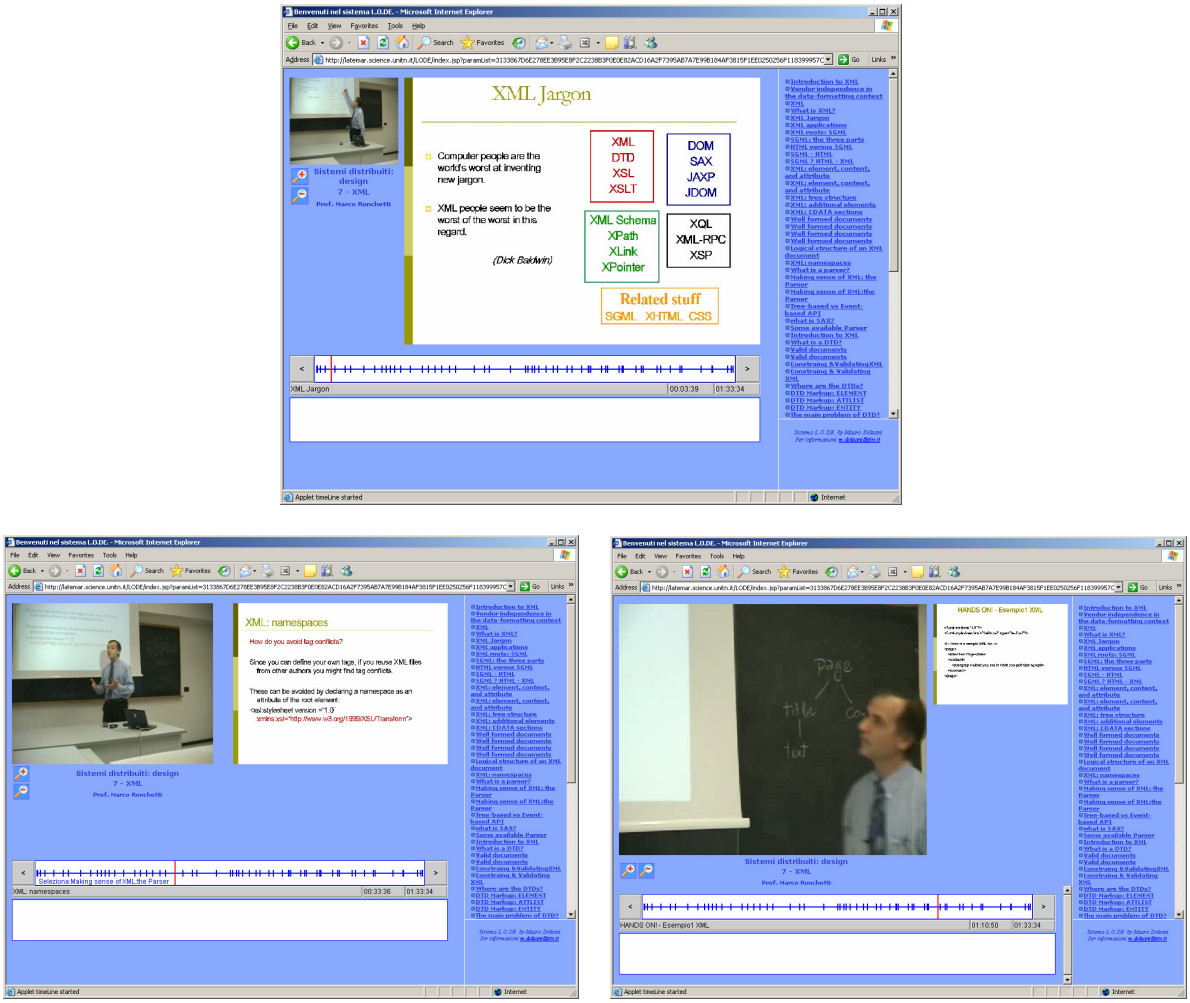

Figure 3: *The LODE system showing video at three different resolutions.*

The system is composed by several subsystems: OnStage, SlideConverter, Encoder, Publisher, WebServer, VideoStreamer, DataStorage, ServiceBroker. The logical flow among the subsystems is pictorially shown in figure 4, and it will described in the rest of this section.

OnStage is the application that resides on the laptop. All the other services could reside on a single machine, or on a battery of distributed PC's to allow scaling the system. SlideConverter and Publisher are Java-RMI based services that are made available on the network. Encoder is a deamon waiting receive on a socket data that need to be encoded. WebServer and VideoStreamer are the servers accessed by the web clients. DataStorage contains the DataBase where all the auxiliary data are made persistent. ServiceBroker allows to keep track and find the machines on which the services are made available. For instance, several users accessing at the same time the streaming service could lead to network congestion. Therefore it might be necessary to distribute instances of the streaming server in several locations. Also, the videos require a large amount of disk space. When encoded in an MPEG4 file at the quality level we need, an hour lecture requires approximately 100 Mbytes. A 50 hour course need therefore 5 Gbytes (which by the way happens to be approximately the size of a DVD: a fortunate coincidence). We are currently planning to put on line the whole set of courses of the 5 years of the Computer Science basic and advanced curricula, for a total of roughly 50 courses: 250 Gbytes of space. Such space allocation must be doubled since for each course we need to put on line the version usable as a stream, and a version suitable for download that can be

copied on CD/DVD by students. It might be necessary to partition the whole storage on more than one machine. The Service Broker (not shown in Fig.4) allows to make distribution and relocation of services fully transparent.

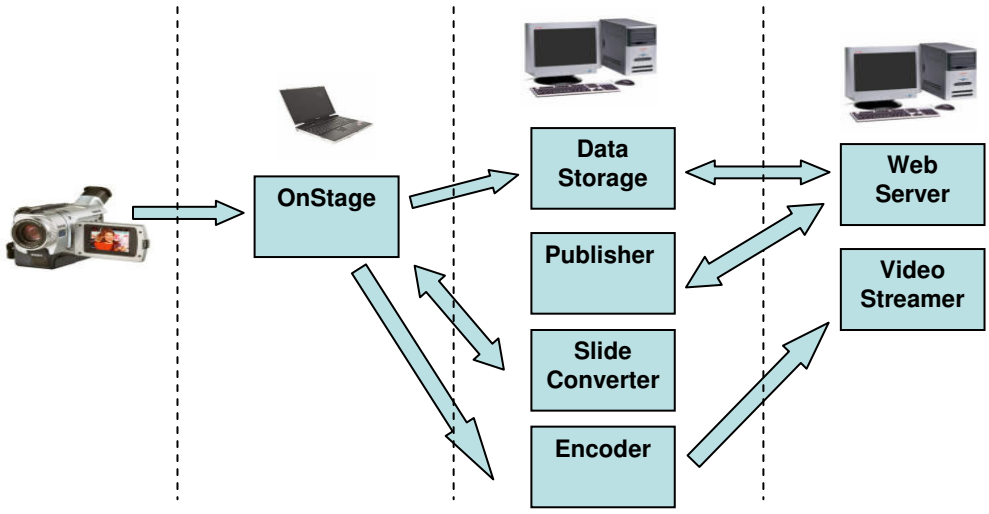

Figure 4: *The logical flow among the subsystems*

The operator that acquires the lecture uses the OnStage application. S/he needs to create a new lecture in a course, and to associate one or more presenters to the lecture. Then s/he must associate material to the lecture. Typical material is made of the slides that the speaker is going to present, converted in jpeg files, although in principle it could be any kind of file that can be rendered by a web browser. For each element (file) a title must be provided, that will appear in the list of topics. Conversion of a PowerPoint presentation in a set of jpeg files and automatic extraction of the title of the slide is available as a Java RMI service provided by the SlideConversion subsystem. These data are saved on persistent storage. The conversion is matter of seconds, so it is sufficient that the operator gets a copy of the PowerPoint presentation just two minutes before the start of the lecture. After loading the material, the operator sees two main panels: in one the images coming from the camcorder are presented, in the second a list of the titles is shown. Whenever the speaker shows a new document, the operator just needs to click on the corresponding title, and the document gets associated to the right time of the acquired video stream. All the operator has to do is to make sure that the camera gets the desired image (pointing it correctly and operating its zoom), check that the images and the sound are correctly acquired, and select the right document. Since in general it is not necessary to move the camcorder a lot, we observed that these operations do not require a high cognitive load, so that it is reasonably feasible for a student to follow the lecture while at the same time acquiring the video. During the process the OnStage gets a video stream through the DV channel and splits it in two copies. One copy is used to show on the laptop screen the acquired images, the other copy is used as input for an internal encoder that produces on the flight a high-quality low-compression AVI file containing a MPEG-4 video and an MP3 audio stream. The AVI file is saved on disk. The associated documents, as well as the temporal information that allow to associate the right document to any given time, are saved on persistent storage. During the process the operator can suspend and resume the acquisition of the stream at any time, and the temporal information is adjusted accordingly.

At the end of the lecture the OnStage process invokes the Encoder subsystem, and passes the AVI file to it. Encoder processes it and produces an MEG4 file compressed at a higher rate. The resulting MPEG4 file is then published on the streaming server. A link to the newly generated lecture is published on a web page. When the user clicks on the corresponding link, a servlet get the necessary data from persistent storage, and includes a link to the suitable resource on the streaming server.
Publisher has the duty to prepare the downloadable version of the lecture. It simulates a user's request to the web server, and creates a static version of the HTML pages with the embedded video that the web server sends as response. Then it prepares a zipped version that can be used to download a whole lecture and produce a CD, and publishes it on the web server.

An important point is that both Encoder and Publisher can run unsupervised. In such way, we've been able to cut down the operator time needed to publish a lecture to a total time that is literally few minutes more than the lecture itself. Since this has an impact on running costs of the system, we consider this to be an important achievement.

4. Implementation notes

We decided to base our system on the reuse of building blocks either existing as free software, or available for free. We used Java for development of the glue, services and interfaces. We could not find platform-agnostic building blocks. After examining what we could find for the Windows and for the Linux environments, we decided that tools available for Linux fitted better our requirements. Therefore LODE was developed as a Linux-based system (though the different subsystems could in principle live on different operating systems). We are still examining the possibility of writing a Windows version of OnStage, mostly because a Linux environment could be unfamiliar for many people, which contrasts with our requirement that the system should be usable with a very short training. In the Linux arena, we found particularly useful three building blocks: FFmpeg (http://ffmpeg.sourceforge.net), the Quasar DV codec – libdv (http://libdv.sourceforge.net/), and Mpeg4IP (http://mpeg4ip.sourceforge.net/).
Quasar is a software codec for DV video, the encoding format used by most digital camcorders. Libdv was developed according to the official standards for DV video: IEC 61834 and SMPTE 314M. We found especially useful the application playdv that allows playing a raw DV stream on the video of the PC, and permits to reduce the CPU load by reducing the reproduction quality. We use it within OnStage only to monitor the acquisition process: a low quality in this phase is not a problem, and the quality of the saved stream is anyhow not affected.
FFmpeg is a fast video and audio converter, that can grab from a live audio/video source. Several parameters can be modified to adjust the quality/compression of the encoded streams. It gets a raw DV streams and produces an AVI file containing a MPEG-4 video and an MP3 audio stream. We use it within OnStage to encode at high quality, and within our Encoder subsystem to decode it.
The Encoder subsystem re-encodes the stream at lower quality as an MPEG4 file readable from QuickTime. Besides this, Encoder implements a queue so that more lectures can be submitted for conversion without waiting the end of active conversion processes. The reason why we pass through the double encoding process is that saving the original DV stream would be impossible (it would require too much disk space). On the other hand, direct encoding at higher compression would require too much CPU time. Moreover it is useful to be able to permanently save a high quality copy that can be used for video editing, or for re-encoding at a different compression rate.
The second encoding process (the one that takes place within the Encoder subsystem) is done by using Mpeg4IP. This software includes many existing open source packages and the "glue" to integrate them together. It is a tool for streaming video and audio that is standards-oriented and free from proprietary protocols and extensions. In particular, we use xvidenc to generate the new MPEG4 video stream and faac to generate the AAC audio stream. Finally, by using mp4creator the .MP4 file is created.

The video streamer we choose is Darwin Streaming Server (http://developer.apple.com/darwin/projects/streaming/), the open source version of Apple's QuickTime Streaming Server technology that allows to send streaming media to clients across the Internet using the industry standard RTP and RTSP protocols.
As a web server we chose Apache Jakarta Tomcat (http://jakarta.apache.org/tomcat/): the servlet/JSP container that is used in the official Reference Implementation for the JavaServlets and JavaServer Pages technologies. Although a standard web server would be enough, we preferred Tomcat because JSP make it easier to integrate with a Learning Management System.

The Learning Management System we're using was built in house. It is called eLeaf and has a set of standard features, comprising the typical e-learning tools and environments. It integrates a module for giving some support to mobile users [6]. It was developed on a three-tier architecture, with the business logic embedded in Enterprise Java Beans and the presentation logic controlled by a combination of Struts and Tiles. Integration between LODE and eLeaf can be achieved in different ways. The first option is to integrate LODE into eLeaf (sharing the same web application "war" file) so that access to the pages generated by LODE can be controlled through the eLeaf permission checking layer. Although such solution is highly satisfactory from the point of view of eLeaf, it binds LODE to this particular LMS. A second solution is more general, although it requires the ability to add an active module (e.g. a servlet) to a generic LMS. Such module should dynamically generate a URL that contains a single parameter generated by encrypting together a time stamp and the data used by LODE (i.e. identification of course and lecture), and should be configured in such way that it can be called only from within the LMS itself. In such way, implicit user identification is achieved by embedding links to this servlet in pages whose permissions are

checked by the LMS. When the user clicks on such link, the control servlet is activated on the LMS side, the dynamic parameter is generated and the request is forwarded to LODE. On the LODE side, a servlet only accepts requests coming from a specified machine (the one hosting the LMS), decrypts the parameter and checks the timestamp. If this is valid, it allows loading the lecture.

4. Conclusions.

LODE was built to harvest our experience in making traditional frontal lectures available asynchronously through the Internet, while supporting also those students who do not have a fast Internet connection at home. It implements all the requirements that we gathered while using a similar system over one and a half year. In particular, it allows extending the model to lectures that are not based on PowerPoint presentations, since it offer a video that has a large enough resolution to show the writing on a blackboard. Its acquisition module runs on low cost equipment based on a laptop connected to a digital camcorder and to a radio microphone. The post-processing and publishing phase are fully automated. The system was designed with special attention to minimize running costs and maximize ease of use. LODE is in production since September 2004.

References.

[1] Haataja Arto, Kontkanen Sirpa, Suhonen Jarkko, Sutinen Erkki “Teaching University-Level Computer Science to High School Students over the Web” ED-MEDIA 2001.

[2] The Point Topic Business DSL Report, published by Point Topic Ltd, Cardinal Tower, 12 Farringdon Road, London EC1M 3HS, United Kingdom (www.point-topic.com)

[3] Baecker, R.M., Moore, G., Keating, D., and Zijdemans, A., (2003) Reinventing the Lecture: Webcasting made Interactive, Proceedings of HCI International 2003, Crete (Greece), June 22-27 2003.

[4] Ronchetti M., "Using the Web for diffusing multimedia lectures: a case study." Proceedings of the "ED-MEDIA 2003" Conference, Honolulu, Hawaii, USA, June 23-28, 2003, p.337

[5] Ronchetti M., "Has the time come for using video-based lectures over the Internet? A Test-case report". Proceedings of the IASTED International Conference "Computers and Advanced Technology in Education 2003", Rhodes (Greece), June 30 - July 2, 2003, p. 305

[6] Colazzo, L., Molinari, A. and Ronchetti, M. “Integrating Mobile Devices in a multi-platform Learning Management System using web services”, submitted to ED-MEDIA 2005.