Vrije Universiteit Brussel

Faculty of Science
Department of Computer Science
and Applied Informatics

SynergE-Learning: A PaperPoint-Based Lecture Capturing and Annotation Tool

Graduation thesis submitted in partial fulfillment of the
requirements for the degree of Master in Applied Informatics

Tom Antjon

Promotor: Prof. Dr. Beat Signer

SEPTEMBER 2011

Vrije Universiteit Brussel

Faculteit Wetenschappen
Departement Computerwetenschappen
en Toegepaste Informatica

SynergE-Learning: A PaperPoint-Based Lecture Capturing and Annotation Tool

Proefschrift ingediend met het oog op het behalen
van de graad van Master in de Toegepaste Informatica

Tom Antjon

Promotor: Prof. Dr. Beat Signer

SEPTEMBER 2011

Contents

Abstract

Nowadays, we get in contact with presentations on a daily basis. They show up in business meetings, at conferences and even for school lectures, presentations are used to assist the speaker. In most cases, PowerPoint is used for giving these presentations, but instead of assisting the speaker, the software often limits the possibilities to convey the real message. Listing facts by bullet points, being restricted by the size of a slide, being forced to use a linear way of presenting or being confined behind the computer, are only a few of the shortcomings of current presentation software.

In a school environment, these limitations cause a depersonalised feeling to students, where it should be more important to challenge students and keep them active. In an attempt to solve these issues, several alternative presentation tools have been proposed, each providing a different approach in presentation style, from predefined paths through slides to free-flow presenting. Lecture capturing tools have also been introduced to give back a part of the interaction to the student.

PaperPoint is a paper-based presentation tool that enhances existing software to overcome some of the issues mentioned above. By using a digital pen and specialised paper, it becomes possible to navigate through the presentation via printed handouts or add annotations to slides.

In this thesis we have built the SynergE-Learning Web Portal, a website that can read the logfiles made from the input devices used during a PaperPoint session. This lecture capturing tool is able to re-enact whole presentations from logfiles, making it easy for students to review a recorded presentation, view a missed lecture or add their own annotations to slides and share them. It is a lightweight tool in the way that it only needs a logfile, a presentation and optionally a video or audio stream. The recording of lectures will allow the students more flexibility during classes and allow them to review it afterwards, making annotations and sharing their knowledge.

To enhance the student's educational experience, the portal allows students to personalise their own settings for reviewing lectures. To achieve this goal, several slide views can be chosen, different annotation methods are provided and several interaction methods are available. We make students the centre of attention again, giving them the opportunity and flexibility to take control of their own education.

Abstract

Vandaag de dag komen we dagelijks in contact met presentaties. Ze worden gebruikt voor zakelijke ontmoetingen, op conferenties en zelfs tijdens de lessen op school worden presentaties gebruikt om te spreker te ondersteunen. In de meeste gevallen wordt er steeds naar PowerPoint gegrepen om presentaties te geven. Maar in tegenstelling tot de spreker te ondersteunen, kan PowerPoint vaak de spreker tegenwerken in het overbrengen van de juiste boodschap. Het opsommen van feiten, beperkt zijn door de slide afmetingen, gedwongen worden om een lineaire presentatiemethode te gebruiken of gedwongen worden voor de computer te blijven, dit zijn maar enkele tekortkomingen van de huidige presentatie software.

In een leeromgeving zorgen deze beperkingen voor een gedepersonaliseerd gevoel voor de studenten. Nogthans is het steeds belangrijker geworden om de studenten zo actief mogelijk te houden en te blijven uitdagen. Een aantal alternatieve presentatie programma's proberen deze tekortkomingen aan te pakken, elk met hun eigen visie. Dit gaat van vooropgestelde paden tot een volledig vrije navigatie tussen slides. Naast nieuwe presentatie technieken is ook het opnemen van lessen geïntroduceerd om het depersonaliseren tegen te gaan.

Ook PaperPoint, een op papier gebaseerd presentatie programma dat bovenop bestaande presentatie software draait, voorziet functionaliteiten om bovenstaande tekortkomingen op te lossen. Aan de hand van een digitale pen en gespecialiseerd papier is het mogelijk om vanop afstand te navigeren doorheen een presentatie doormiddel van uitgeprinte handouts. Deze handouts kunnen ook gebruikt worden om nota's te maken.

In deze thesis introduceren we het SynergE-Learning Web Portal, een website die logboeken, gemaakt via PaperPoint, kan analyseren. Dit stelt de software instaat om een hele presentatie herop te bouwen en studenten de mogelijkheid te geven om een les te herbekijken. Dit kan handig zijn om gemiste lessen in te halen, oudere lessen herop te frissen of lessen van aantekeningen te voorzien en deze te delen met medestudenten. Omdat het web portaal enkel een PaperPoint logboek en de overeenkomstige presentatie nodig heeft om een les herop te bouwen, is dit een heel licht systeem. Optioneel kan er nog een audio of video bestand aan de presentatie toegevoegd worden. De mogelijkheid om presentaties te herbekijken, maakt het voor studenten mogelijk om tijdens de les aandachtiger te zijn en niet steeds te moeten noteren. Het kunnen herbekijken van lessen, aantekeningen maken en aantekeningen delen maakt de leerervaring meer interactief.

Om studenten nog meer flexibiliteit te gunnen, is het mogelijk om hun profiel volledig te personaliseren. Om dit te realiseren hebben we verschillende overzichten gecreeërd van lesinhouden, we voorzien ook verschillende methodes om nota's te maken alsook verschillende visualisatietechnieken voor deze notities. We zetten de student terug in het middelpunt van de belangstelling, we geven hen alle mogelijkheden om hun eigen educatie in handen te nemen.

Acknowledgements

E-learning has become a hot topic in the educational environment. Constantly, new technologies are introduced to improve the learning experience of students. I have always had a large interest in education and ICT, and this thesis allowed me to come in contact with new presentation techniques and state of the art educational software that combines those two interests.

Also having a teaching degree, has me interested in the effects of ICT in a school environment on the performances of the student. By making this thesis, I was able to do research on these topics and hopefully this thesis will contribute to making eduction interesting and personal again.

Hereby I want to acknowledge everyone who contributed to this thesis. First of all, I would like to thank my parents for giving me the opportunity to fully discover my capabilities and for giving me the support that I have needed during these years. They were always there for me during both study as personal issues.

I would also like to thank my promoter, Prof. Dr. Beat Signer. The weekly meetings pushed me to keep working forward. He was always prepared to give me advice, listen to my ideas and guide me in the right direction. I would also like to thank him for proofreading my thesis and improving it week after week.

I would also like to thank Matthew Aucella, Vice President of Sales at Tegrity, for providing us with a fully functional testing environment of Tegrity and several statistics of the universities that incorporate Tegrity into their curricula.

Finally, I would also like to thank my friends for their support and my father for proofreading my thesis.

Chapter 1

Introduction

"School Sucks" is a quote of Dr. Tae, one of the persons who is challenging the current educational system, because he and many others believe that the traditional school system is outdated. More and more students grow tired of school and one has to wonder what the reason is. At universities, class rooms are packed with students following lectures mostly supported by PowerPoint presentations.

This method of teaching provides a very depersonalised approach. Students are not actively attracted to the lecture and lose interest, especially if slide handouts are provided before the presentation has even begun. Edward R. Tufte states in [19] that PowerPoint is presenter-oriented and not content- and audience-oriented. Nevertheless, for non-serious presentations where the cognitive style is not that important, PowerPoint remains a very useful presentation tool. However, the more content-critical a presentation becomes, the more PowerPoint can harm the content rather than enhance it. In those cases it might be more efficient to abandon PowerPoint and use presentations based on technical reports or specialised analytical software.

The following list presents some of the main issues of existing presentation tools.

- Most presentation tools only provide a linear navigation style which limits the freedom of the presenter;
- they restrict users to the slide format, forcing them to use bullet points, abbreviations and short sentences;
- they restrict the mobility of presenters, obligating them to return to the computer for navigating, making annotations or making remarks.

Figure 1.1 also shows that the "PowerPoint Phluff", as Tufte calls the graphical animations, transitions and other "entertaining" functions, can harm a presentation. The graph in this picture represents the data of the table that is shown next to it. It is clear that the table is much more informative than the graph. The graph may seem to be more sophisticated, but it is actually less clear and hides more data rather than enhancing it.

Estimates of relative survival rates, by cancer site[1,2]

	% survival rates and their standard errors			
	5 year	10 year	15 year	20 year
Prostate	98.8 0.4	95.2 0.9	87.1 1.7	81.1 3.0
Thyroid	96.0 0.8	95.8 1.2	94.0 1.6	95.4 2.1
Testis	94.7 1.1	94.0 1.3	91.1 1.8	88.2 2.3
Melanomas	89.0 0.8	86.7 1.1	83.5 1.5	82.8 1.9
Breast	86.4 0.4	78.3 0.6	71.3 0.7	65.0 1.0
Hodgkin's disease	85.1 1.7	79.8 2.0	73.8 2.4	67.1 2.8
Corpus uteri, uterus	84.3 1.0	83.2 1.3	80.8 1.7	79.2 2.0
Urinary, bladder	82.1 1.0	76.2 1.4	70.3 1.9	67.9 2.4
Cervix, uteri	70.5 1.6	64.1 1.8	62.8 2.1	60.0 2.4
Larynx	68.8 2.1	56.7 2.5	45.8 2.8	37.8 3.1
Rectum	62.6 1.2	55.2 1.4	51.8 1.8	49.2 2.3
Kidney, renal pelvis	61.8 1.3	54.4 1.6	49.8 2.0	47.3 2.6

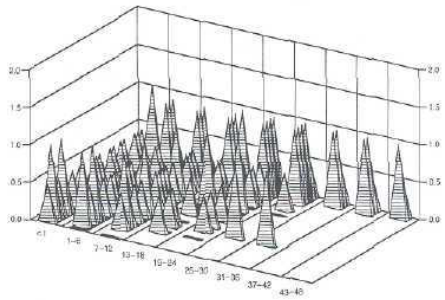

Figure 1.1: Table vs Chartjunk: Extract from *The Cognitive Style of PowerPoint* [19]

Some of these issues have already been addressed by other presentation tools. They introduce new and innovative ways of navigating, breaking the boundaries of the 'slide' concept and moving away from the use of bullet points. Nonetheless, people still prefer the use of PowerPoint. PaperPoint, by Signer and Norrie [16], is a tool that enhances PowerPoint and other presentation tools with additional features including paper-based navigation and annotation services to address the limitations of those presentation tools.

All of the proposed solutions do make presentations more content-oriented. Lecture capturing tools have become a method for schools to make lectures audience-oriented. Currently existing capturing software provides students with the flexibility to review lectures. However, in most cases, it stays with only reviewing it. Lectures are recorded and can be replayed. Only a few solutions provide students with the option to make annotations and when they do, it is restricted to textual annotation based on time events. This is only one of the shortcomings. The following list gives an overview of useful functionalities that are not available in existing software:

- Provide various presentations overviews: linear overview, path visualisation or single-instance slide overview;
- Provide various methods to annotate: annotate a time event, annotate a slide section or annotate an annotation;
- Provide different annotation content: text, image or audio/video;
- Provide flexible sharing methods for annotations: private, share with person, share with group or share publicly;
- Provide various annotation visualisation methods: list view, timeline visualisation or on-slide visualisation;

The purpose of this thesis was to provide a portal website where students can review presentations delivered via PaperPoint. In a first phase, we enabled the portal to replay the presentation by reading the logfiles of PaperPoint, it visualised an automatic navigation through the slides with the adding of the annotations made during the presentation. Afterwards we extended the portal, so that users can also add their own annotations afterwards. These annotations can be personal, directed to the

presenter to provide feedback or to ask additional information or shared with the subscribers of the course for discussion or questions. The portal has been implemented in a flexible way so that users can personalise their learning experience. For this purpose, several views for annotating slides are provided as well as different ways to navigate through the slides. The goal of this thesis was to give back flexibility to the students by making lectures more interactive and activating the students, which enhances the learning experience. Also the presenters can profit from this system. The annotations can give an indication where there are problems, it allows them to understand their audience better and to improve the overall presenting skills. It also brings the audience and the presenter closer together, in a more interactive environment.

In chapter 2 we first have a look at PowerPoint and some of the flaws that Tufte mentioned. Some alternative presentation tools are then introduced, giving a short overview of the tools and their take on "presenting done right". Next, the PaperPoint tool will be covered more extensively as this thesis builds on the technology presented by it.

The idea of recording and replaying lectures with lecture capturing software will be elaborated on in Section 2.2. Some of the existing solutions will be introduced, presenting their approach and working points. A comparison of the products will round up this chapter.

Chapter 3 presents the SynergE-Learning Web Portal, starting with the idea behind the system and a short explanation of the used technologies. In Chapter 4, we continue with the implementation of the portal, presenting the different ways to navigate through a presentation. The second major contribution of the portal is the annotation service that it provides. This part will explain how annotations can be created, shared and visualised. Furthermore, the user management of the portal, allowing users to personalise their account, will be explained. In Chapter 5, we go back to the stated issues of current lecture capturing tools and review how they are addressed by the SynergE-Learning Web Portal. Chapter 6 concludes this thesis with some conclusions and ideas for future work.

Chapter 2

Background

In this chapter we take a look at existing solutions. The first section discusses several presentation tools and their benefits and shortcomings. The second and third section introduces the lecture capturing concept and some existing tools. The chapter concludes with a comparison of existing lecture capturing tools.

2.1 Presentation Software

2.1.1 MS PowerPoint

MS PowerPoint is the most widely used presentation software in the world. Microsoft estimates that there are more than 30 million PowerPoint presentations produced each day. However, Tufte states the following about standard PowerPoint presentations:

"A foreshortening of evidence and thought, low spacial resolution, an intensely hierarchical single-path structure as the model for organizing every type of content, breaking up narratives and data into slides and minimal fragments, rapid temporal sequencing of thin information rather than focused spacial analysis, conspicuous chartjunk and PP Phluff, branding of slides with logotypes, a preoccupation with format not content, incompetent designs for data graphics and tables, and a smirky commercialism that turns information into a sales pitch and presenters into marketeers." [19]

PowerPoint has a decent approach for making presentations, otherwise it would not have become this popular. However, there are several restrictions to it that prevent it from being the perfect presentation tool.

Tufte uses the risk assessment of the Columbia shuttle as an example. After a piece of foam insulation broke of during liftoff and hit the left wing, hereby damaging the thermal protection, there was a need for a quick and smart analysis of this problem within a limited timespan, as there were only 12 days left before Columbia would re-enter the atmosphere. There were 3 reports prepared, consisting of 28 PowerPoint slides, dealing with the debris impact. The first problem he mentions is the excessive use of bullet points, the lower-level bullets often mentioned doubts and uncertainties, but the high level ones were quite optimistic. In most cases, the low-level bullets undermine the big executive ones, however they are presented far less important by the use of the bullet point mechanism. Secondly PowerPoint, and several other presentation tools, provide low resolution presentation formats that encourage vague references, abbreviations and short sentences because there is not enough room for specific and precise statements. A third flaw in the presentation was that PowerPoint is just not good enough for presenting mathematical and scientific facts. The software is not equipped for

such subjects. In this case a technical report would have been the better approach. It could have given a much more detailed analysis of the problem, stating facts instead of using the pitch-style that PowerPoint provides.

The more important and technical a presentation gets, the less appropriate PowerPoint becomes. Therefore, other and more specialised tools should be used such as technical reports or specialised analytical programs. The presentation tool should be chosen accordingly to the content of the presentation.

For general presentations, MS PowerPoint is still a great product, but specific presentations ask for specific tools.

2.1.2 Prezi

Prezi [11] is an online presentation tool that breaks the linearity of presentation slides, being the standard in presentation tools such as PowerPoint. It does this in a very unique way. The tool is a map that serves as a storyboard where all ideas are spread on. Users can work in various levels by zooming in and out. This way a topic can be put on the map and by zooming in, details and subtopics can be revealed. Zooming out on the other hand can present the bigger picture or can serve as a transition between topics.

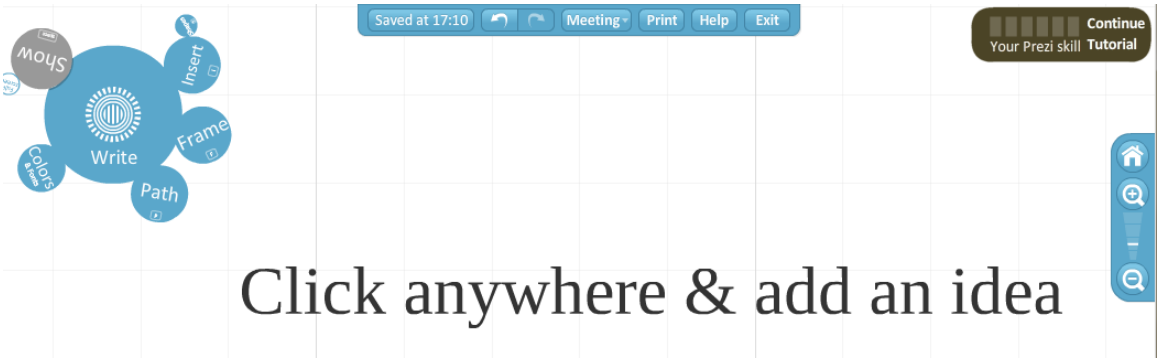

Figure 2.1: Prezi's storyboard

Figure 2.1 shows the storyboard of Prezi. On the map, all topics can be laid out and by using the zooming tools on the right-hand side, the working level can be changed to add more details. Once all topics are constructed on the map, a path can be defined how the presentation should run. This can be done by switching to path mode, using the option buttons on the left. By selecting content in a particular order, a linear path is created. This approach can be useful for new adopters of the system. More experienced users can use a free-flow mode, where they can just navigate throughout the map.

2.1.3 FLY

A planar interface for authoring presentations, this sums up what FLY [4] actually is. FLY works in a similar way as Prezi as it uses a canvas to map out ideas instead of boxing them into slides. The technology is based on mind-mapping where similar topics are grouped together, which is a more natural way of structuring content.

When creating a presentation, the user maps the ideas over the canvas and groups them according to a topic, this creates gestalt objects. By taking snapshots of the presentation at different zooming levels, users can create a path for navigation through the topics. Many paths can be defined, where each path can represent a certain presentation.

Figure 2.2: FLY

Users can also navigate freely, discarding a path, which can be useful for answering questions from the audience.

2.1.4 PaperPoint

PaperPoint [16, 12] is a tool that builds on technologies for interactive paper to enable presentation software to be controlled from printed slide handouts. The tool also enables the user to easily annotate the slides via the handouts and can be extended with third-party implementations.

This system is based on the interactive paper framework, iPaper [14]. iPaper is a plug-in for iServer, which is an information server specialised in cross-media annotation and link services [17, 15]. The framework enables the definition of active areas within documents which can be linked to digital information or services.

As MS PowerPoint is currently the most commonly used presentation tool, PaperPoint builds on this technology to provide its services. However, as the system makes use of a generic presentation interface, it can easily work with other presentation tools as well. Using PaperPoint requires little to no extra effort. Users can prepare their presentations in PowerPoint as they are used to. Afterwards the presentation will go through the publication process shown in Figure 2.3. In a first phase the process fetches all the shape information from the iPaper/iServer databases and creates a new empty document. Based on the static information associated with shapes, the publishing engine loads the required resources from the file system or downloads any content from the Web and inserts it at the shape's exact position. Afterwards, any variable content is retrieved and inserted into the document. In the case of PaperPoint, the content source consists of a single PowerPoint document. The publishing tool extracts the variable content page by page from the PowerPoint document. The single slides are resized to the slide shape's dimension and added to the paginated document. After all the static and

variable content has been inserted into the document, the Anoto pattern is added. Finally, the newly created interactive document is printed and used to interact with the PaperPoint application. While currently the digital pen technology of Anoto [1] is used, which uses a camera and a specialised pattern on printed paper to calculate the position of the pen, the system is flexible enough to work with other pen tracking tools.

The handouts that are created with PaperPoint allow presenters to navigate through the slides during a presentation. They keep the handout with them at all time and interact with them via the digital pen. This also provides presenters with more freedom. Whereas normally a presentation has a fixed path, and presenters should know the whole presentation layout by heart, the PaperPoint handouts provide an overview of the slides. This way, presenters know exactly which slide will be shown next. Additionally they can change the path of the presentation on-the-go. Presenters can decide to skip certain slides, go back to previously shown slides or change the order of the topics that will be handled during the presentation. This functionality provided by PaperPoint enables non-linear presentations with existing presentation tools.

During the presentation, the PaperPoint server will act upon the inputs from the pen. The augmented handouts provide navigational buttons on top that can be activated with the pen. Furthermore, the user can annotate the slides in a very natural way by drawing and writing on the handouts. The system also implemented a concept called active components. These are areas that can be defined on the paper. These areas can be linked to a piece of Java code that can be executed on either the server or client side. This allows developers to extend the framework without having detailed knowledge of the framework itself.

The use of the PaperPoint handouts to navigate through the presentation allows an easier overview of the slides, non-linear navigation, dynamic annotation and it also re-enables mobility for the presenter.

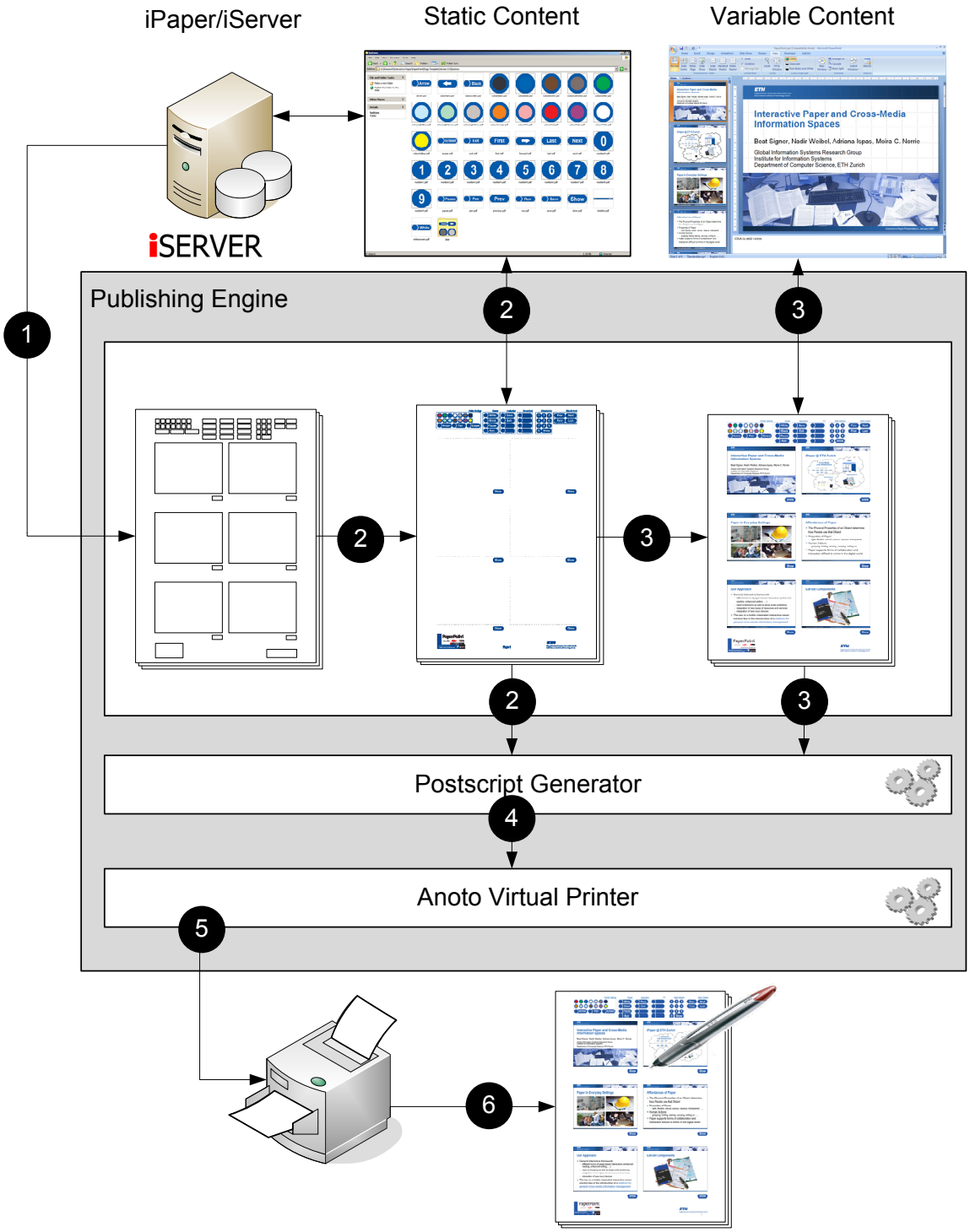

Figure 2.3: PaperPoint printing process

2.2 Lecture Capturing

2.2.1 Effects of Lecture Capturing

The emerging use of technology in education, over the last decade, has lead to a shift in the educational approach. Where lectures used to be very teacher-focussed, nowadays more and more focus lies on the students being the centre of education. This shift can be explained by a number of changes:

- Technology has become more common, can be easily acquired, and the overall knowledge of it is far better than 10 years ago.
- The technology has been accepted within classrooms; laptops are regularly used by students.
- The group of students has become more heterogeneous. Part-time students, work-students, full-time students of different ages are all mixed in a one single class group.
- The educational system has stepped away from fixed student programs by implementing the ECTS system. Personal programs have become more common, where students can compose their own curriculum mixing courses or only enrolling for a course to obtain a particular credit.

All these reasons ask for more flexible ways of education. Lecture capturing provides such a flexibility, however many different forms apply. In the most rudimentary form, lectures are recorded and can be replayed afterwards. Today, the lecture capturing system has changed to where the focus lies: on personalisation. Students should be able to define how they will review lectures and how they will access that content.

In the sense of this personalisation, students should be able to annotate lectures and create bookmarks to attract attention. Those bookmarks can help the learning experience. They can be shared with other students, rising questions about content or they can be used to ask for assistance on difficult parts. By making students take a teacher's role, they can enhance their own knowledge on a subject and share their point of view on content.

Not only students can benefit from those bookmarks, also teachers can be supported by these as they provide feedback on parts of the course. A teacher can reform parts of a course where a lot of difficulties occur or add additional information to the lecture via the bookmarks.

All these methods serve the purpose of making the viewing of a lecture more active so that it enriches the traditional way of education. Many would question the lecture capturing solution since they might keep away students from the class rooms. Studies show however that this is not the case. Students see the reviewing of lecturing as an added part of the education, helping them to understand lectures better. Reviewing the lectures afterwards can help students to fully focus in the classroom, instead of having to divide their attention to either listening to the lecturer or to making notes. On occasions that students cannot make it to class, they can review the lecture afterwards as well. This also paves the way to online courses.

Lecture capturing enables efficient, personalised educational solutions. It fits perfectly with making the education system more flexible and shifts the experience to the students themselves.

2.3 Lecture Capturing Tools

As a part of our research, we evaluated a number of lecture capturing tools. In this evaluation, the main focus was to establish where the products differ, what the advantages and disadvantages are and what the shortcomings of the software are so that our implementation could address these points. In [5], ELI notes the most used lecture tools that are on the market nowadays. Two of these tools were tested. The people of Tegrity provided us with a test environment in which we could evaluate their product to the fullest.

2.3.1 Tegrity

Tegrity is a web-based program that revolves around a main website where students and lecturers can log in. Functions differ according to the user's role.

The Lecturer View

In case of a lecturer, the main view shows the courses assigned to that person. The button 'Record a Class' gives quick access to the recording tool, which gets installed on the first use. After that first installation, the tool can also be accessed directly without having to go via the website.

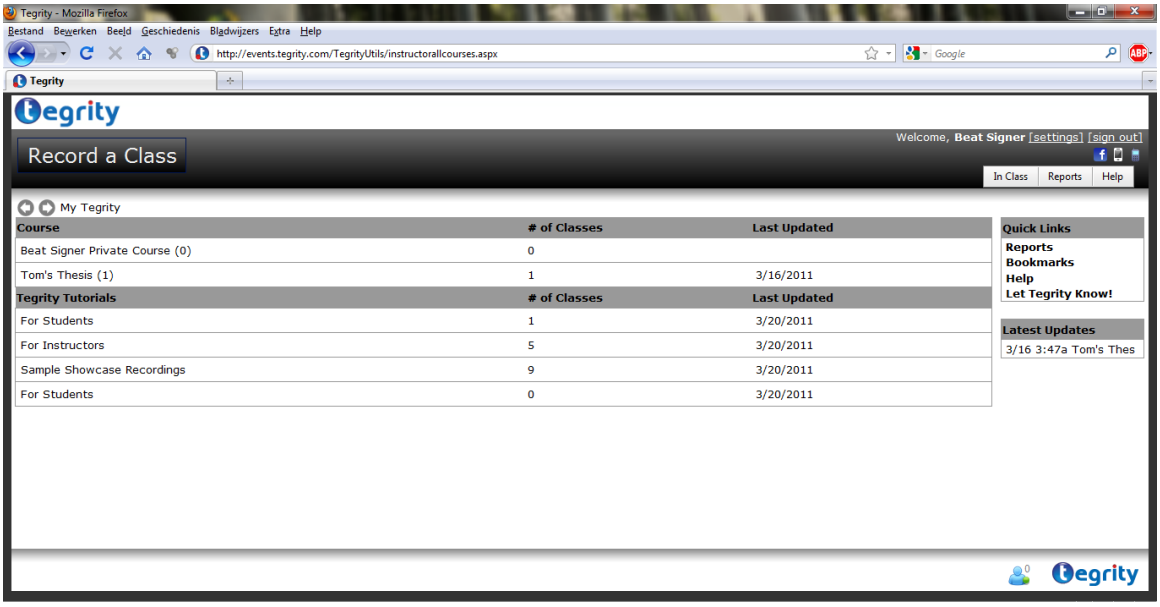

Figure 2.4: The main view of a lecturer

In the recording tool some preferences can be set: selecting whether or not the instructor should be recorded and, if so, which video source should be used, selecting the audio source, testing the audio/video source, selecting to which course the recording should be uploaded, etc.

Once the 'Record a Class' button has been clicked, Tegrity starts with the capturing of all the actions that are performed on the PC. If the audio/video option is enabled, it will also record those channels. The recording can be paused/resumed at any time via the control bar, which also holds a button to end the recording. Once ended, Tegrity offers the possibility to preview the recording, delete it or upload the recording to the Tegrity website to the selected course. While the recording is being uploaded, the Tegrity system will process the video, create chapters and index the content so that the search function can be used on it.

After uploading the recording, it can be edited via the website. Also additional files and links can be added to it. Now the students can access the recording and replay the recorded lecture. Figure 2.5 shows the recording tool that gets installed on the first use.

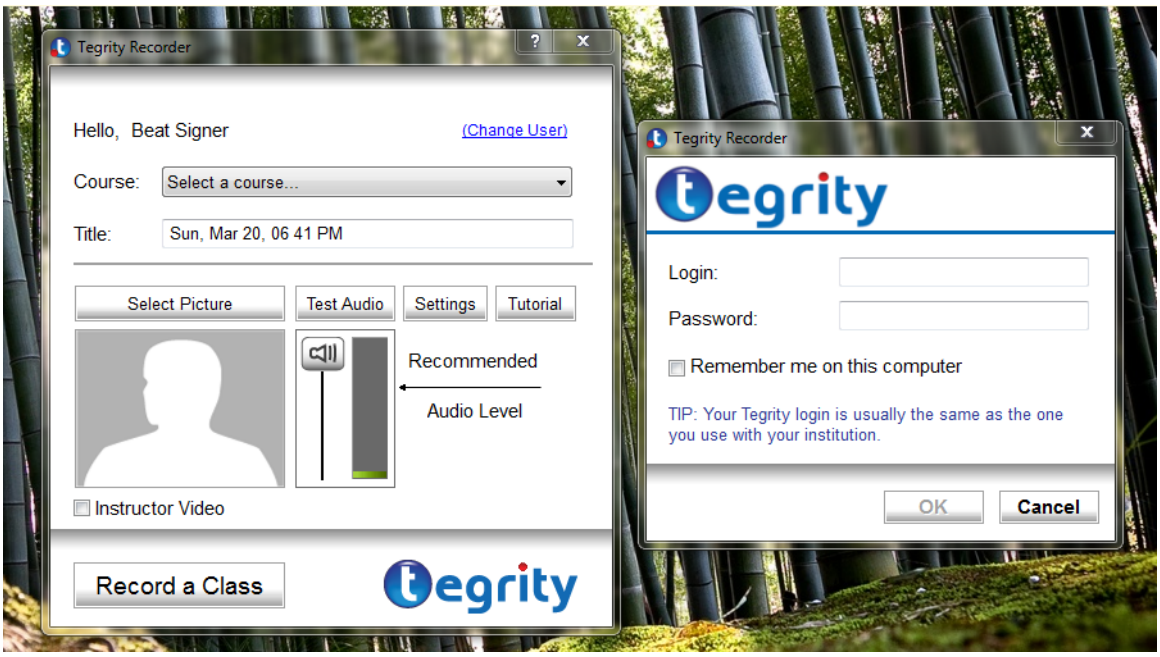

Figure 2.5: The recording tool

The Students View

When students are logged in on the Tegrity website, they get a similar view as the main view of the lecturer. Students get an overview of all the courses that they have been enrolled to, each containing class recordings. This view is shown in Figure 2.6.

Once a course has been selected, the student can choose a class he wants to replay or he can download or review some additional files that have been added by the lecturer. Each class contains an overview of the recording, showing thumbnails of the several chapters. A chapter covers a certain change of screen, as long as the recorded screen stays the same. For example, if a PowerPoint slide is shown, everything recorded in during that time is seen as one chapter. When the screen changes, for example because there is a slide transition, a new chapter is created.

The recording can be started from any of the chapters, and once started, an extra thumbnail is provided that keeps track till where the replay has been viewed so that it can be resumed from that point in a later session.

From this view of the program, the student can also download the audio or video podcast of the recorded class. This way the student can transfer them to mobile devices and view them there. Also a chat function is provided to the students enrolled in the same course. This way, they can discuss a lecture, ask questions about certain topics and exchange ideas.

The Replay Window

For the actual replaying of the lecture, a new window will open and start playing the recording from the chosen chapter on the previous screen. The replay window, shown in Figure 2.7, consists of several

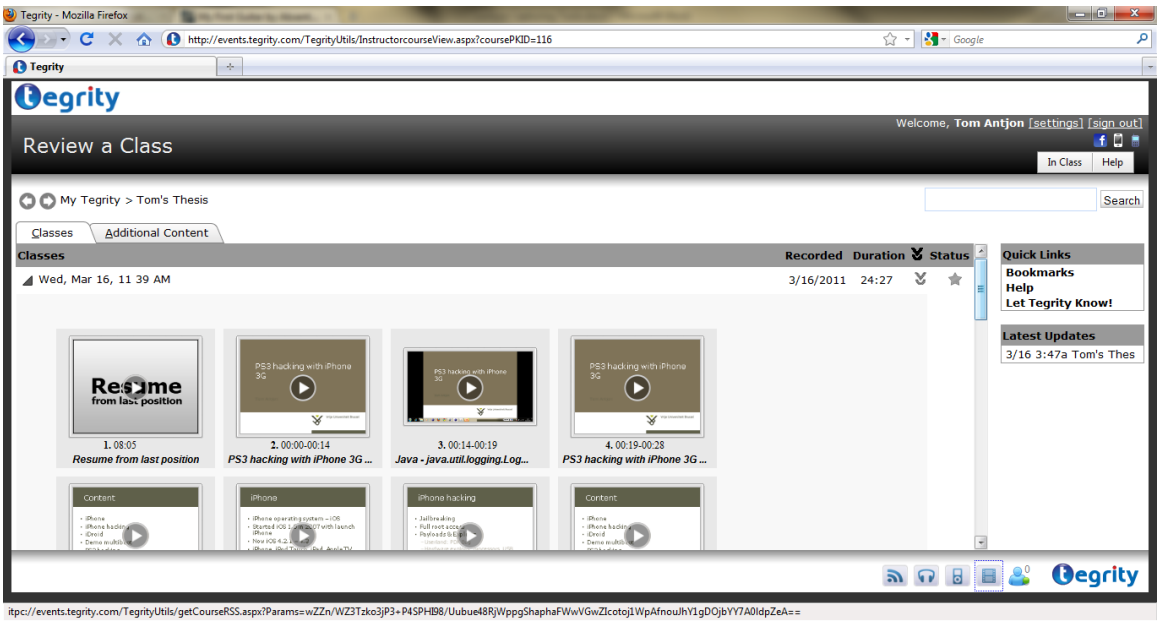

Figure 2.6: Class overview

panels: on the left-hand side of the window the instructor video is shown, below this panel there is also a bookmark area.

In this area bookmarks can be added or previously added bookmarks can be consulted. Next to adding bookmarks via the Tegrity website, there is also a possibility to add bookmarks via the Tegrity iPhone app. Bookmarks can be marked as 'important', 'unclear' or 'comment' and an annotation can be added to it. The bookmarks can then be shared via email or Twitter. The right-hand side of the replaying window contains the screen capturing that has been done during the recording. This panel is the most important part and receives the most attention of the window.

Below all these panels is the navigation bar, which shows a timeline of the recording with some regular media controls like play and pause. The student can also call the chapter overview, print the chapters or download the recording.

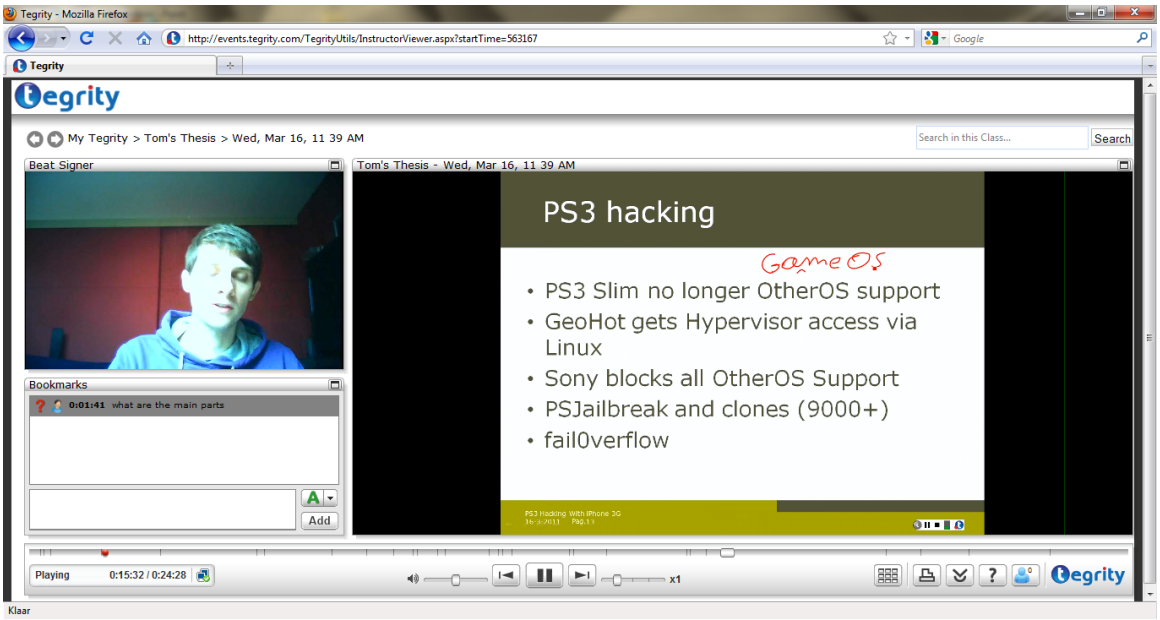

Figure 2.7: Replaying a recorded class

Replaying lectures is the main focus of Tegrity. However, there are some additional options that can be enabled as well. For example, the students can also be allowed to record and publish their own recordings. The lecturer can also distribute tests via the Tegrity site. While the students take their test, every action is recorded and can be reviewed by the lecturer.

2.3.2 Panopto

Panopto, the second lecture capturing tool that is discussed, also provides the option to request a testing environment. Because the VUB already owns a licence to CourseCast, Panopto would not grant us this testing environment. Thus the evaluation of the tool can only be done from the viewer's side, not from the presenter's side, and from other reviews [10].

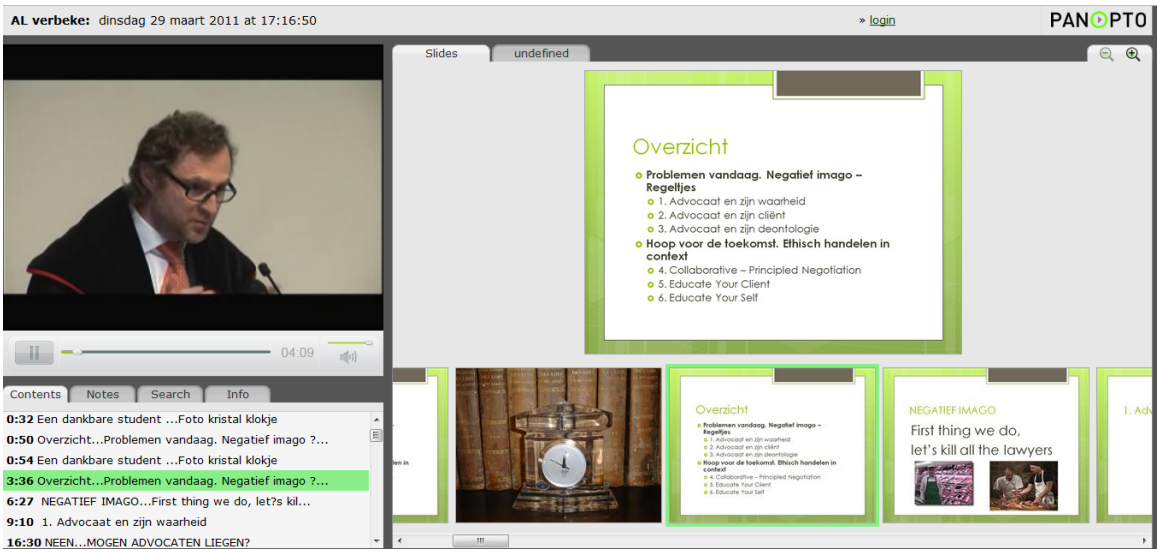

Figure 2.8: Panopto

Panopto does things differently than Tegrity. Figure 2.8 shows the Panopto replay window. We can see that the provided timeline is part of the video controller and that the slide associated with the current video timestamp is shown next to the video. However, the timeline has no indication to where the slide transitions occur. An overview of the slides is listed below the video, showing the slide title together with the timestamp on which a slide transition took place. Navigation can thus be achieved either via the video controller, via the transition list below the video or via a linear visualisation of the slides that can be found at the bottom of the screen.

Like Tegrity, slides can also be annotated. However, this is visualised in no other than an annotation list as a separate tab page under the video. An annotation consists of a timestamp and a plain text annotation. Other users can comment on it but nothing else has been implemented. The annotations are linked to a channel. These channels can be private, for example for personal notes, or they can be posted in public groups.

It is clear that Tegrity and Panopto have a similar approach in presenting the recording. However, Tegrity takes interaction a step further with more flexibility for annotations and chat functions. Furthermore, both solutions also provide options to download a podcast version of the lecture. This enables the students to review lectures on mobile devices. These podcast versions are created when the recording is processed. In the processing phase, the chapters are also created and the slide content gets indexed for search options.

2.3.3 Elluminate

Elluminate[1] is a web-based screen capturing tool. It evolved its system from a business web-meeting and collaboration tool to a system for an educational platform. The system captures the screen recordings made by a teacher, but the students can interact and share their own screen as well. This tool is more likely to be used for on-line distance learning and less for recording in-class lectures. The tool shows the transitions of the screen and visualises it on a timeline. Students can make textual notes, chat with other class members and interact on a shared whiteboard.

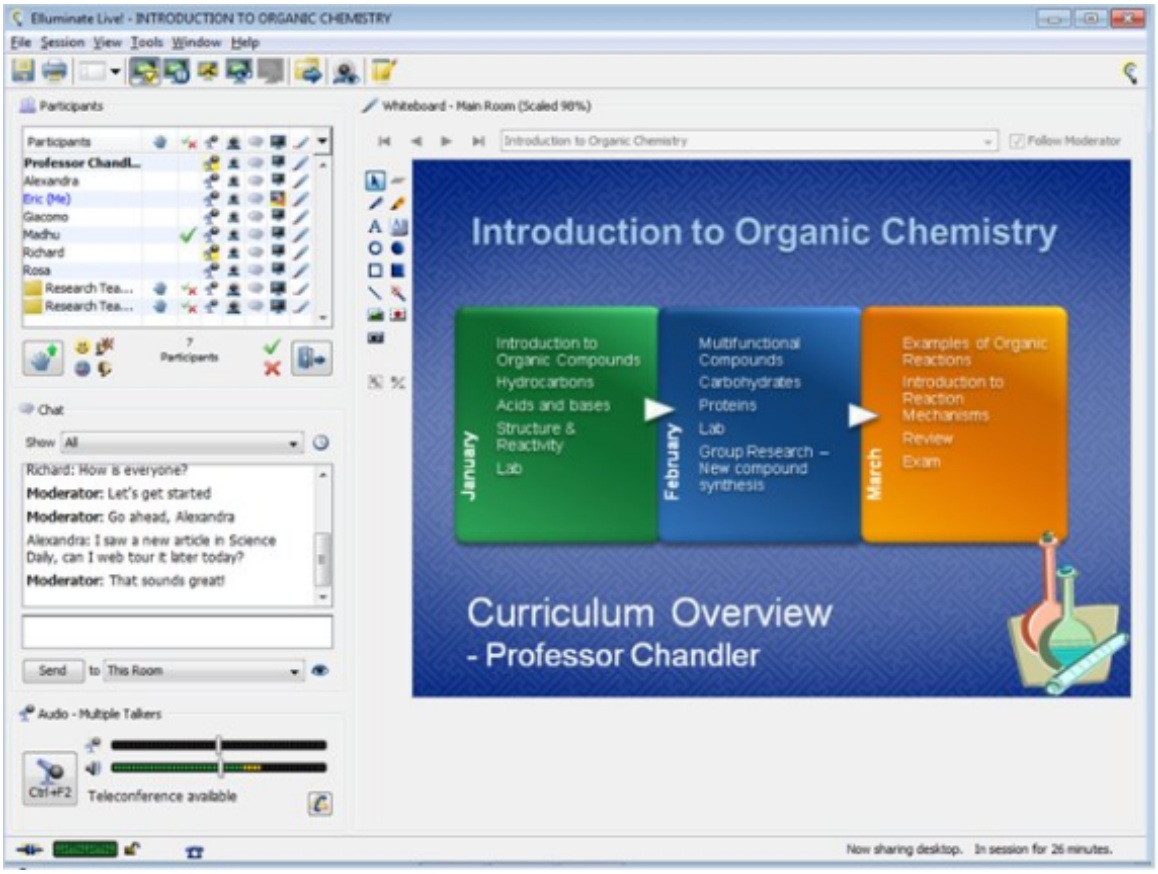

Figure 2.9: Elluminate

Figure 2.9 shows the Elluminate tool as a live-broadcast. The main window serves as an interactive whiteboard where lecturers can project their presentation. Teachers can make annotations in real-time and, if the permission for a collaborative whiteboard is enabled, students can make annotations as well. When the broadcast is replayed however, there is no possibility for making annotations. Also there are no indications on the timeline, when and where annotations were made. In replay mode there is an additional timeline, and media control buttons as shown in Figure 2.10. The markings on the timeline, which indicate the screen transitions, and a list of slide numbers are the only two ways to visualise a lecture overview.

Figure 2.10: Elluminate media controller

The main shortcoming to this tool is that it is more focused on live broadcasting. There is hardly any focus on post-lecture actions besides the replay of the broadcast. When broadcasting live

[1] `http://www.elluminate.com/`

however, there are various methods that contribute to an interactive lecture. Students can collaborate on a shared whiteboard, they can share their own desktop or use a web cam for direct interaction. Furthermore, a chat function is integrated or a quiz can be set up. All these functions focus on live-broadcasting and have a business-oriented origin.

Echo360

Echo360's blended learning environment[2] is an all-in-one solution that plans, records and publishes instructional content for students to review anytime and anywhere. It specialises more in large scale recordings in classes that are equipped with video/audio hardware. For this Echo360 provides Safe-Capture. This is a hardware module that can be added into the school's server room and be integrated with the A/V hardware. However, Echo360 also provides screen capturing tools or integration with the instructor's laptop.

This system offers little to no functionality for interaction. There is no option to annotate the slides nor is it possible to add any form of feedback to the system. Other implementations offer a chat function or options to make a poll or quiz, but Echo360 offers no interaction options at all. It does however give a clear overview of the different scenes of the lecture. A linear overview shows thumbnails of the different slides that are used during the presentation. Furthermore, the software provides a search option for the lecture and the ability to share a link to lecture via Facebook or Twitter. Figure 2.11 shows how the students can review lectures.

This system requires a central server for managing the capturing tool. This solution would be the choice for large scale implementations.

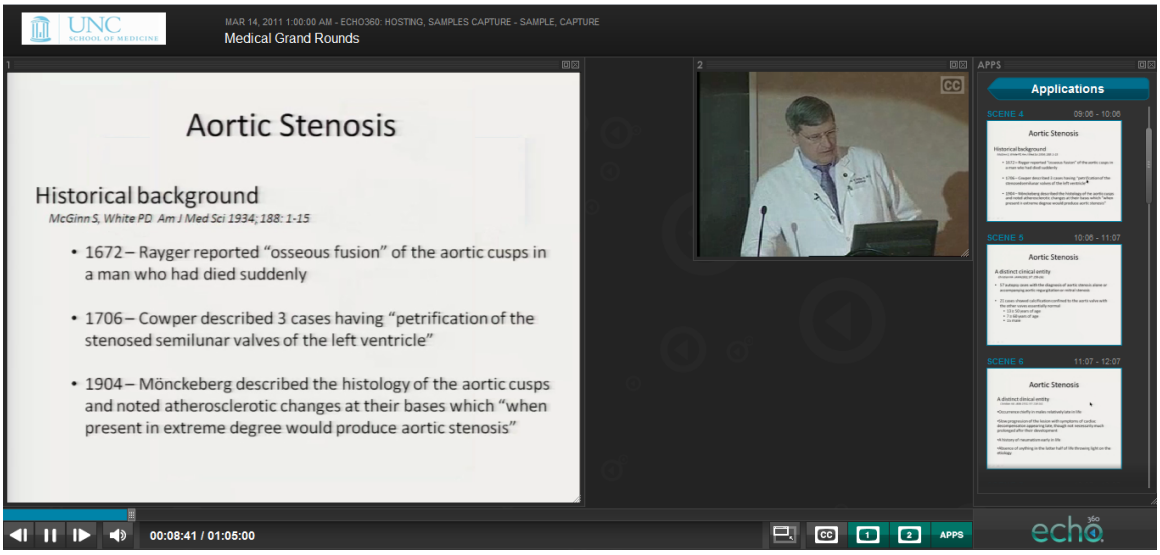

Figure 2.11: Echo360 EchoSystem player

Sonic Foundry Media Site

Media Site, a product of Sonic Foundry[3], is another learning capturing tool. It is similar to Echo360 as it also uses specialised hardware for the recording of lectures. However, there is no implementation to use the other video or audio devices that are integrated at the client side.

[2] `http://echo360.com/`
[3] `http://www.sonicfoundry.com/mediasite/`

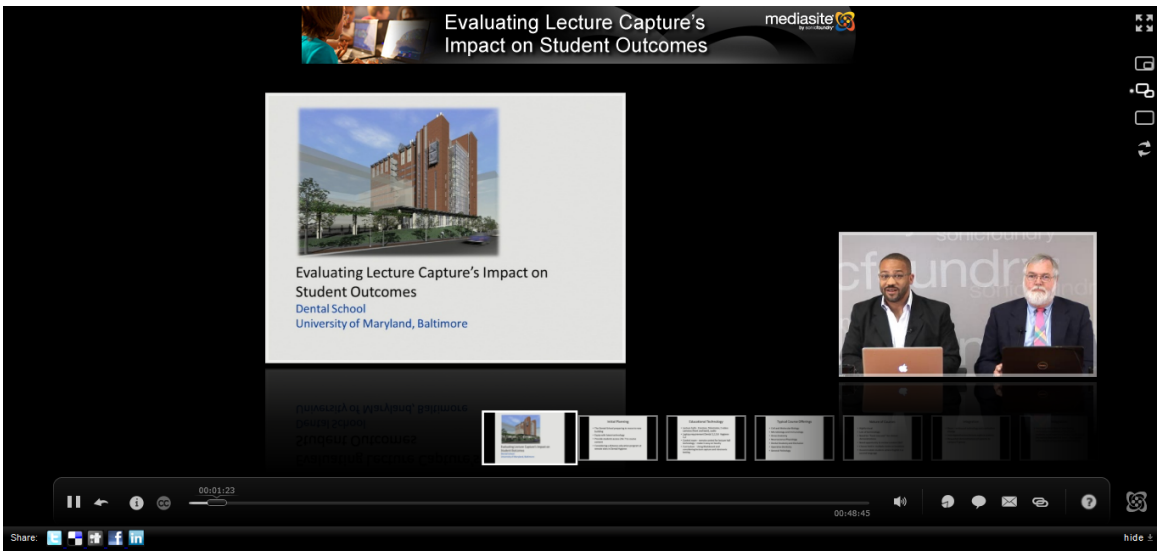

Figure 2.12: Sonic Foundry Media Site player

This solution also lacks the ability to annotate recordings. On the other hand, there is an option to interact via a question panel. During a live presentation, the viewer can use this option to interact with the presenter. Figure 2.12 shows the Media Site player where we can see the interaction buttons in the lower right corner. The speech bubble displays the question panel where the viewer can ask a question, the question is accompanied with a timestamp of the broadcast as well. Next to the speech bubble there is another interaction method, namely polls. These polls can, for example, ask the audience what topic should be further elaborated. These options focus on live-broadcasting and more for immediate interaction. As we have seen with other live-broadcasting software, this solution also lacks support for post-lecture annotations and interaction. An overview of the lecture is provided in a similar approach as in Echo360. The overview visualises a linear division of the scenes of the lecture. The Media Site player also shows the presented slide and an audio/video recording of the presenter, which can be switched to shift the focus of the recording. Furthermore, an overview is visualised and at the bottom of the application one can find the buttons for controlling the recording.

Like Elluminate, Sonic Foundry's Media Site, originates from the business perspective and therefore lacks implementations that are specific for educational purposes.

Matterhorn

Matterhorn [9] is an open source solution of the Open-Cast Community[4]. This solution enables schools to install a low cost lecture capturing system. As most introduced lecture capturing tools, this system also captures the screen recording of the presenter. It can also integrate any form of audio/video. This way schools are not obligated to have specialised rooms.

Matterhorn's main focus lies in the recording and redistributing of lectures. The media player of the tool is shown in Figure 2.13. It has an implementation for the basic annotation of slides. However, this a new implementation of the software and there was no specific information about it. The information that is available states that there is a possibility to annotate recordings based on time events. On a certain timestamp, a plain text annotation can be added to the recording. There is no information if these annotations can be shared.

Other than the annotation service, there are no options for interaction. However, the system is

[4] `http://www.opencastproject.org/matterhorn_capture/`

open source so schools are free to adjust the system to their needs. The main advantage over the other implementations is that the chapters of the lecture can be adjusted. In other systems, chapters are linked to screen transitions. With Matterhorn, additional chaptering can be done as well.

Matterhorn is different from all the other implementations. It provides an open source solution built on a versatile and extensible foundation. The use of Matterhorn does ask for experts to set up the system and configure it. There is also the possibility to build extensions for the system and personalise it completely to the desires of the school. However, this will take time and resources. For an out-of-the-box solution, other tools might be more suitable.

Figure 2.13: OpenCast Matterhorn

2.3.4 Discussion

Figure 2.14 shows a comparison of the two lecture capturing tools that we have put to test.

	Tegrity	Panopto
Client (recording workstation) requirements	No external hardware needed Install provided recording app	No external hardware needed Install provided recording app
Server requirements	Hosted solution	Hosted solution/ local server option
Captured content	Capture and playback of video, audio, associating files and links.	Capture and playback of video, audio, associating files and links.
Storage and upload of captured materials	Capture locally and process on central server afterwards	Capture locally and process on central server afterwards
Indexing and searching of content	search anything	only PowerPoint slides
Sectioning of video	Bookmarks + annotations	bookmarks

Figure 2.14: Tegrity vs Panopto

The main advantage of these two solutions is that there is no need for external hardware. The system can be integrated with existing audio and video hardware, only a recording application has to be installed. Both of the introduced products are widely used but lack personalisation options and flexibility.

Most lecture capturing tools follow the same approach: the screen of the presenter is captured and synchronised with video and audio input. Each screen transition represents a chapter and the visualisation of the chapter overview is linear. No attention goes to returning slides; they are just visualised as a new slide. Annotating content also does not have a main focus, only the replaying of the lecture is important. However, giving the students the possibility to annotate can enhance the student's educational experience as well as give feedback to the lecturer. Solutions that do provide the option to annotate content are not flexible. In most cases, only textual annotations can be added and can only be linked to a timestamp. Figure 2.15 shows a comparison table of the discussed lecture capturing tools and their features.

	Tegrity	Panopto	Elluminate	Echo360	Sonic Foundry	Matterhorn
slide overview	linear	linear	✗	linear	linear	linear
transition marks	✓	✓	✓	✗	✗	✓
annotations	basic media	basic notes	✗	✗	✗	basic media
annotation sharing	public/private	public/private	✗	✗	✗	unknown
interaction	chat	question board	chat - whiteboard collaboration	✗	questions and polls	✗
ingest method	screen capture	screen capture	screen capture	screen capture	screen capture	screen capture
process method	on-server processing	on-server processing	on-server processing	on-server processing	on-server processing	on-server processing
conclusion	best experience	replay-oriented	business and webcast-oriented	replay-oriented	business and webcast-oriented	work in progress

Figure 2.15: Feature comparison

This thesis presents a solution that is flexible and customisable by the student so that it can meet their personal choices. The main focus lies on different content visualisations. Student are able to get a much clearer view on the different chapters of a lecture. The second focal point is on annotating content. Students are able to annotate content in various ways. Annotations can be linked to time, space and even other annotations. Different media is supported, students are able to attach documents, video, audio, etc. in addition of the textual annotations that are currently supported by other solutions. The use of PaperPoint enables this kind of flexibility. PaperPoint allows the presenter to give non-linear presentations and also introduces an intuitive way to annotate slides by using a digital pen. All the actions performed in PaperPoint are logged, used to recreate a lecture and this also results in a lightweight solution that can be easily implemented and extended.

Chapter 3

SynergE-Learning Web Portal

This thesis introduces the SynergE-Learning Web Portal for replaying recorded lectures based on the PaperPoint technology. The web portal together with PaperPoint serves as a lightweight lecture capturing tool. In this chapter, we present the functionality of the SynergE-Learning Web Portal. We will have a look at the functionality provided by the web portal and discuss the shortcomings of existing solutions.

3.1 Recording a Lecture

In a first phase of the life cycle, a presenter starts with creating a presentation, for example in PowerPoint. Once the presentation is finished, it can be processed by PaperPoint to create the augmented paper handouts as seen in Section 2.1.4. Presenter can now give a presentation and make use of the PaperPoint features such as remote navigation and annotation. For this, presenters need to start the PaperPoint application on their PC and make a connection between the application and the digital input device. Now all the actions performed with the digital pen will be logged on the PC. It is this file that allows our web portal implementation to recreate the lecture. Figure 3.1 shows the dialogue box that is presented upon starting PaperPoint.

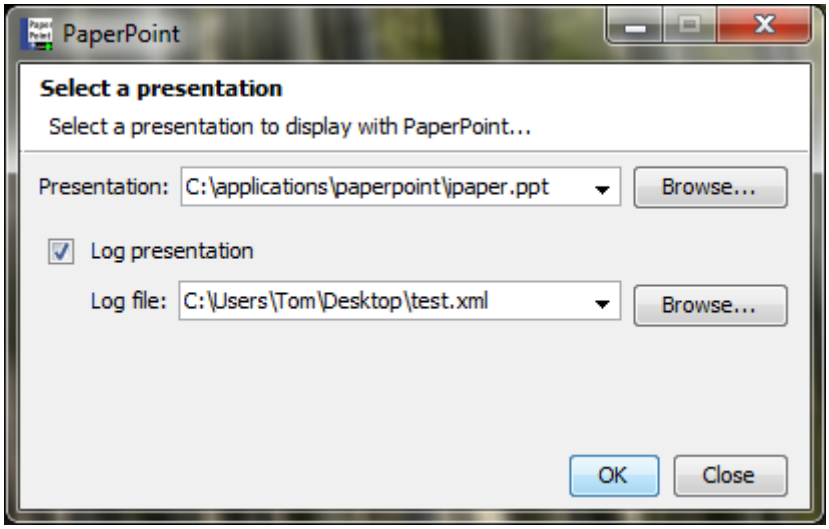

Figure 3.1: Starting a PaperPoint Session

The logging captures, for example, all of the navigational actions between slides. This way the exact moment of slide transitions can be deducted, as well as the slide order. From this information,

we can create an overview of the lecture and show the advantages of non-linearity in presentations that PaperPoint introduces. Next to the navigation, all the annotations made with the pen are captured. This collection of pen data allows us to recreate the annotation and could, in the future, be used to perform OCR on and convert the annotations to actual text. Figure 3.2 shows some examples of the possibilities of PaperPoint.

(a) Handouts (b) Annotations

Figure 3.2: An example of PaperPoint Handouts

After a lecture has been recorded, it can be sent to the web portal and made available to the students for reviewing. For this, the presenter can log into the web portal and create a new lecture. To add a lecture, the presenter needs to upload the presentation itself, the PaperPoint logfile and optionally a audio/video file that was created. Figure 3.3 shows the window that lecturers get when they want to add a new lecture. Here they can define a name for the recording, add it to a course and add the necessary files. The recordings can be added to existing courses or new courses can be added as well. Courses can also be managed via the user settings. In Section 3.4 we will provide more details about the user settings.

Next, the slides of the presentation will be extracted and converted into images and the logfile will be processed. In this phase, every action made by the digital pen during the presentation is analysed and stored in a database. These events are needed to replay the presentation and recreate the annotations that were made. Now that the lecture is created, it can be reviewed by the students.

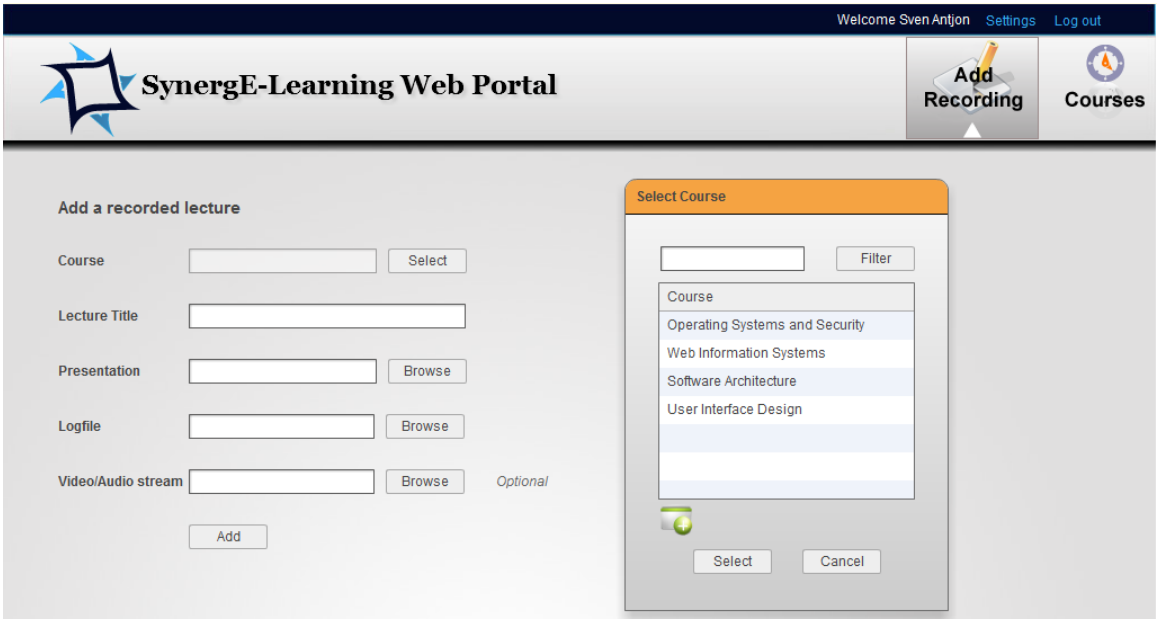

Figure 3.3: Adding a lecture

3.2 Viewing a Lecture

Once a lecture has been selected to be viewed, the viewer will be redirected to the slideshow player of the web portal. Here the user will be able to replay the lecture, including the presented slides and an additional video/audio stream. Furthermore, annotations will be visualised and the user can request an overview of the lecture.

The main view of the web portal focuses on the most important function of the portal, which is displaying recorded lectures. Figure 3.4 shows the replay window. This screen is divided in different panels, each providing a different functionality. The main panel visualises the presented slides, accompanied with a media controller for starting, stopping and pausing the replay of the lecture. The user can also use the timeline to move through the lecture to specific parts. The timeline also provides indications when slide transitions take place. These functions make it easy to navigate flexibly through a lecture. The main panel showing the slides is also accompanied by a smaller panel showing an additional audio/video recording. These two components are synchronised and are both controlled with the media controller.

The timeline of the media controller has been extended with indications to when annotations have been added. The annotations are represented with a colour and a shape, indicating the character of the annotation. The shape indicates the nature of the annotation. The square marks represent timeline annotations. These annotations are created by clicking on a certain timestamp on the timeline and are thus linked to a time event. Next to the square marks, there are also circle marks which linked to on-slide annotations. These annotations are created on by selecting an area on the slide on a certain timestamp. When these events are triggered, the annotation is shown on the slide, indicating the annotation area. The square marks, as well as the circle marks, have a particular colour which indicates the privacy settings of the annotation. The red marks represent private annotations. This means that the annotation is only available for the user who made the annotation or that the creator shared the annotation with a private group. When the mark is blue, however, the annotation is publicly available for every user with access to the course. Figure 3.4 also shows an overview of all

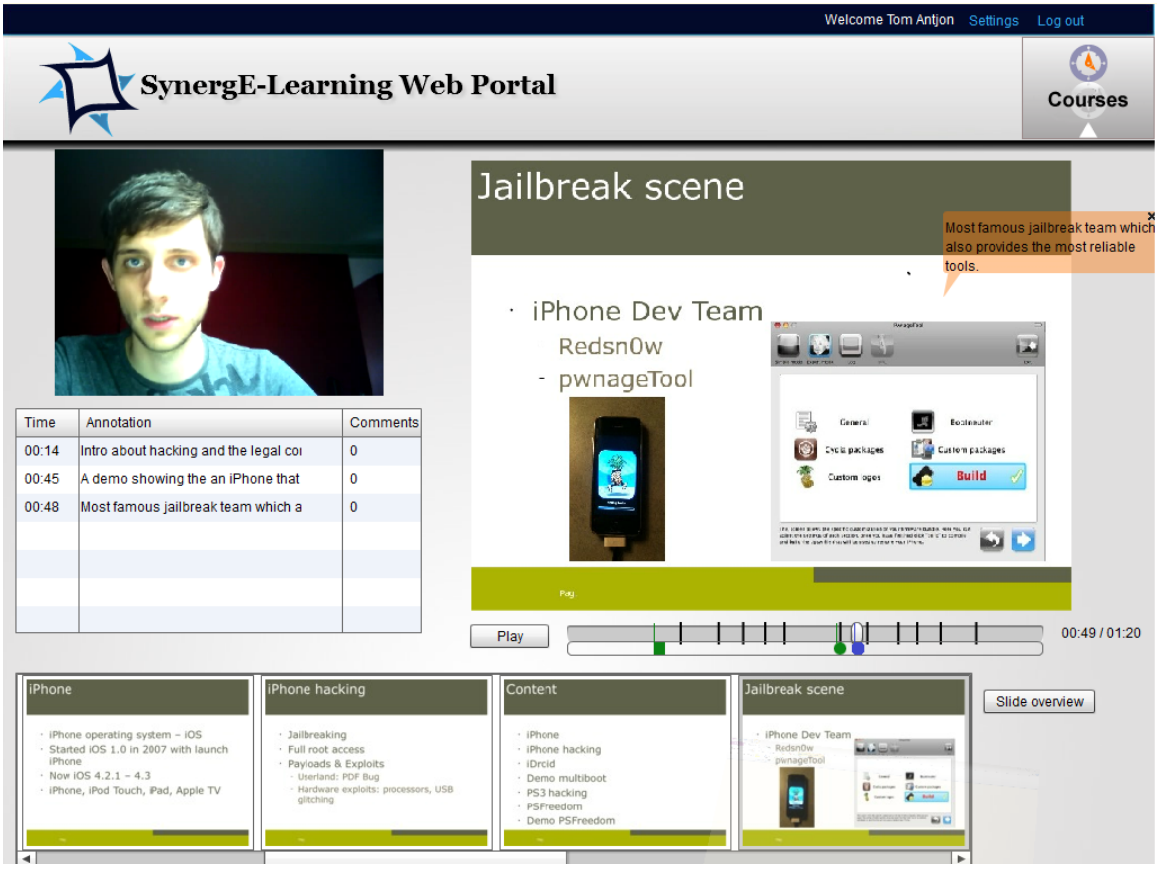

Figure 3.4: Viewing a lecture

the annotations made in the course. They are visualised in a list that can be found below the the video/audio feed. This list provides the possibility to browse quickly through the topics. We will go deeper into the annotation options and visualisations later.

First, we will take a look at the visualisation of the slide overview section. Next to the linear overview that is seen on the bottom of the main view, there are several other overviews that can be consulted by clicking on the 'Slide Overview' button. These options can be useful if a viewer wishes to navigate quickly to a certain slide or would like to see the path of the presentation.

Slide Overview

Most of the existing lecture capturing tools provide an overview of the lecture. They show thumbnails of every screen transition. As most of the solutions are screen recordings, this division of chapters has to be calculated when the recording gets processed. Every time the screen changes, a new chapter is created.

PaperPoint allows a more light-weight solution, as every slide transition is an action made by the digital pen and all actions are logged. By reading this logfile, a far more effective and faster processing is available. It also gives more flexibility to manipulate all the logged actions and additional actions can easily be introduced as well. As the logfile provides information such as the slide number and the order of the presented slides, alternative overviews can be constructed next to the linear one.

Next to the linear path, which happens to be the the traditional way of giving an overview of the slides in existing solutions, we also introduce several other visualisations. One option shows the path of the lecture in a more intuitive way, visualising the non-linearity of the recording. In other solutions the slides are shown in a single row. When slides are visited multiple times in a lecture, they

Figure 3.5: Visualisation of a linear overview

are normally shown in the same row. We introduce an overview that adds a new row whenever the presenter shows a slide that was already presented before. As long as the presenter moves forward through the slides, the overview will stay on the same row. Again, when a presenter goes back to a previously shown slide, a new row is added. This way multiple instances of a slide are positioned below each other and each column represents the number of the slide. Each slide will also have a line indicating the path of traversal. An example of such a non-linear overview is shown in Figure 3.6.

The user can also request a more simplistic overview, where each column represents the slide number and each instance of a slide is on another row, as before, but here the row number represents the instance number of the slide and there is no path defined as well. Each slide in the overview contains a thumbnail of the slide and the timestamp when the instance was shown. An example of this overview is highlighted in Figure 3.7.

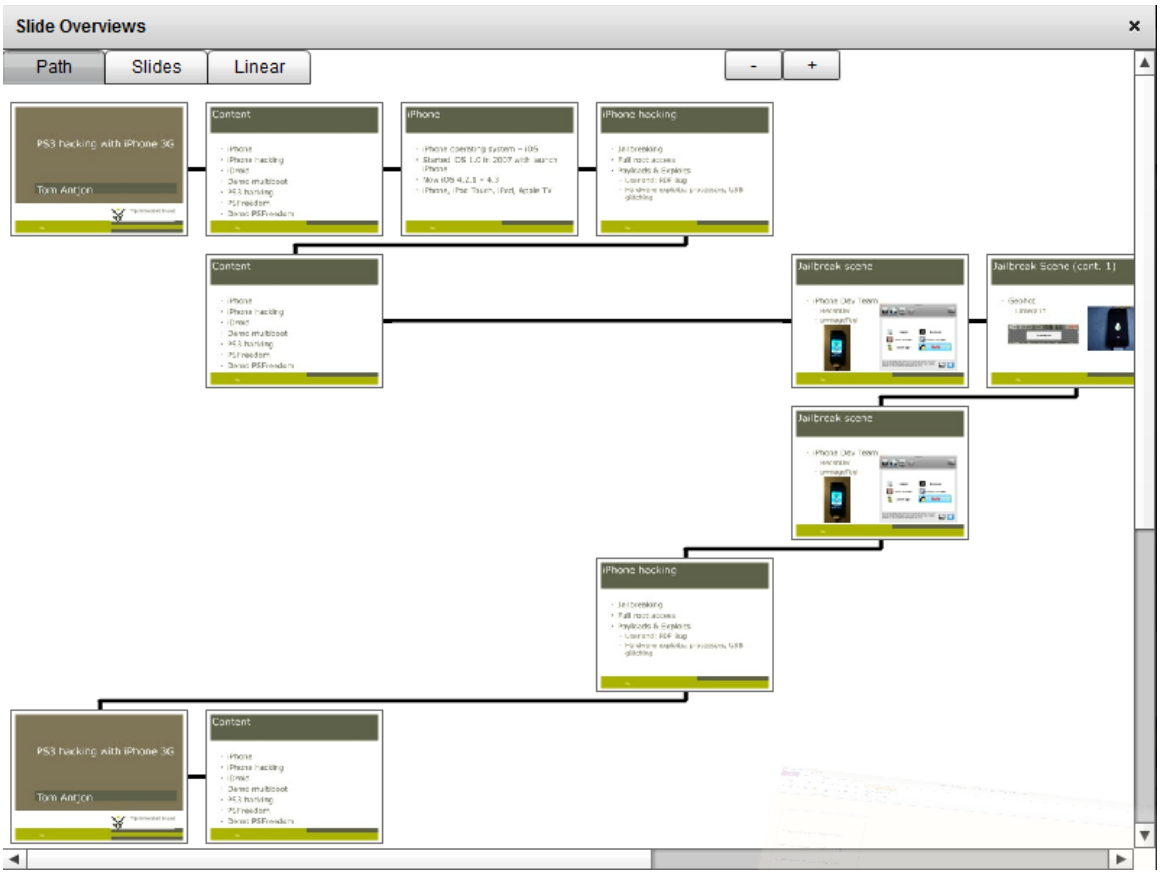

Figure 3.6: An example of a path overview

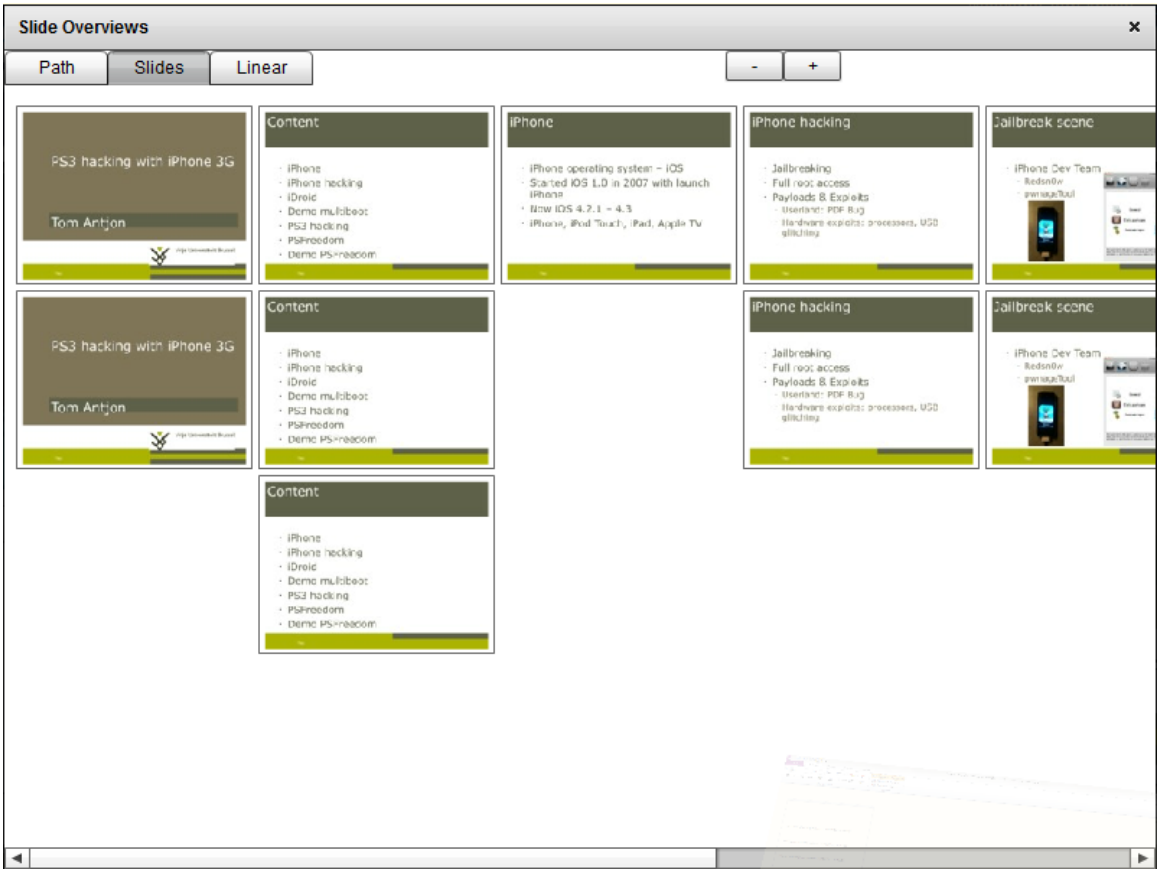

Figure 3.7: An example of the enhanced linear overview

3.3 Annotating a Lecture

The second contribution of this thesis is to provide flexible annotation options for captured lectures. Only a few existing lecture capturing tools support annotating lectures and when they do, it is only in a rudimentary form. The annotation of slides is limited to textual annotations based on time events, with limited sharing options. The overview of these annotations, that is provided by existing tools, is also rather simple. In most cases, the overview is a simple list. However, Tegrity goes a bit further by indicating marks on the timeline as well.

In this thesis, we present the user with several options for viewing the annotations. Before going into the different visualisations, we first need to clarify two different kinds of annotations. On the one hand, we have the annotations that are made by the presenter when the lecture is being recorded. This is done with the PaperPoint technology. The presenter can make notes with the digital pen which are recorded and added automatically when the lecture is being replayed. On the other hand there are the annotations made within the web portal, in most cases by the students themselves. These annotations will be visualised in different ways: on the timeline, on the slides or in an annotation list.

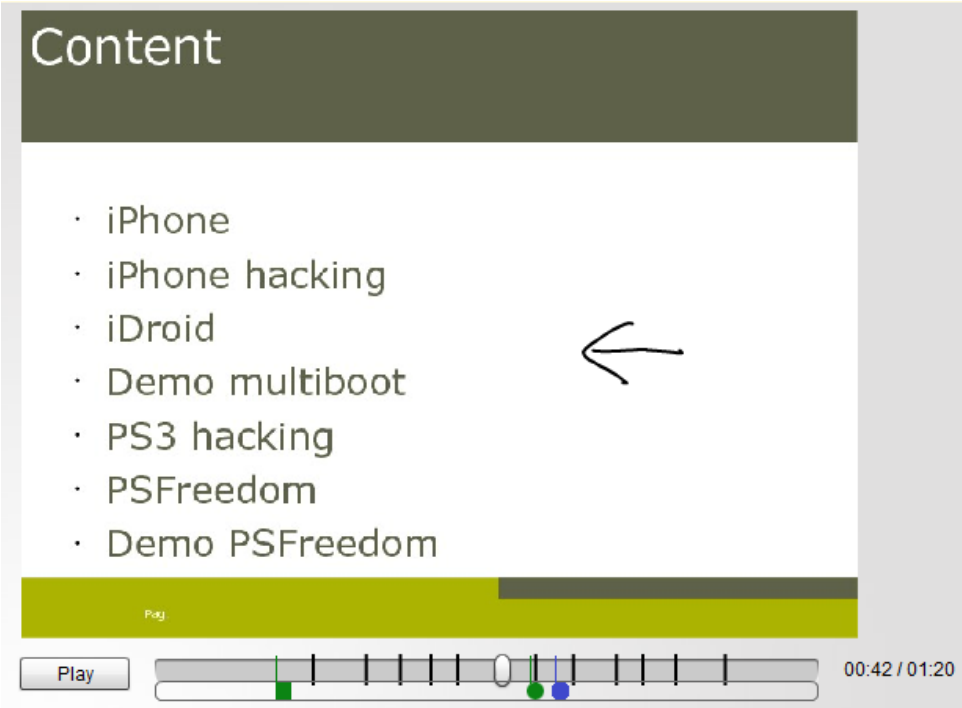

Figure 3.8: An example of a pen annotation on a slide

The time-triggered annotations are visualised as square marks on the timeline and can be accessed by clicking on them. Once clicked, the annotation is shown pausing the replaying of the lecture and replacing the slide show. The annotation is presented on the screen and also shows other annotations that are linked to it. All the annotations are presented in a threaded form as can be seen in Figure 3.9. The timeline also has marks shaped as circles, which represent on-slide annotations. When these marks are triggered, they appear on the slide in a pop-up. However, like the off-slide annotations they can also be presented in a threaded form when clicking on a mark. This way, details of the annotation and the other annotations that are linked to it can also be presented.

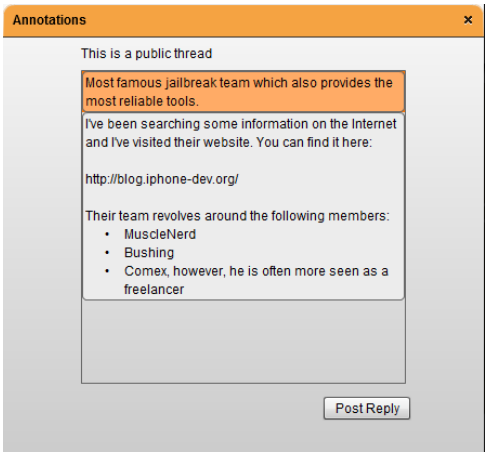

Figure 3.9: The visualisation of an annotation thread

The on-slide annotation is represented as a text bubble pointing to the annotation area of the slide that the user selected upon creation. The text bubbles can be dismissed by clicking the text bubble's close button. Furthermore, the user can also disable the option to let the on-slide annotation pop up. When disabled, the on-slide annotations are handled as off-slide annotations. Figure 3.10 shows an example of some on-slide annotations.

Figure 3.10: An example of a viewer annotation on a slide

A third way to visualise the annotation is via the annotation list, as shown in Figure 3.11. This is a grid view of the annotations with their timestamp and a teaser of the annotation content. It also shows how many replies the annotation received. By selecting an annotation, it can be viewed in the threaded form like the other ones. This list view provides an easy and quick way to access annotations.

Time	Annotation	Comments
00:14	Intro about hacking and the legal coı	0
00:45	A demo showing the an iPhone that	0
00:48	Most famous jailbreak team which a	1
00:49	• Redsn0w sends a RamDisk	0

Figure 3.11: The annotation grid shows a list of annotations

Adding new annotations can also be done in different ways. One can add an annotation by clicking on the timeline so that the annotation will be linked to an associated timestamp. Or the user can click on a certain point at a slide during the replay. this way the annotation will be linked to a certain area on the slide. The on-slide annotation will also be associated with a timestamp.

The annotations can be of various types of media. The most simple form of annotations will be simple text. This can be added from the pop-up window that is displayed once the user has selected a method to add the annotation in the previous paragraph. This window contains a WYSIWYG[1]

[1]What You See Is What You Get

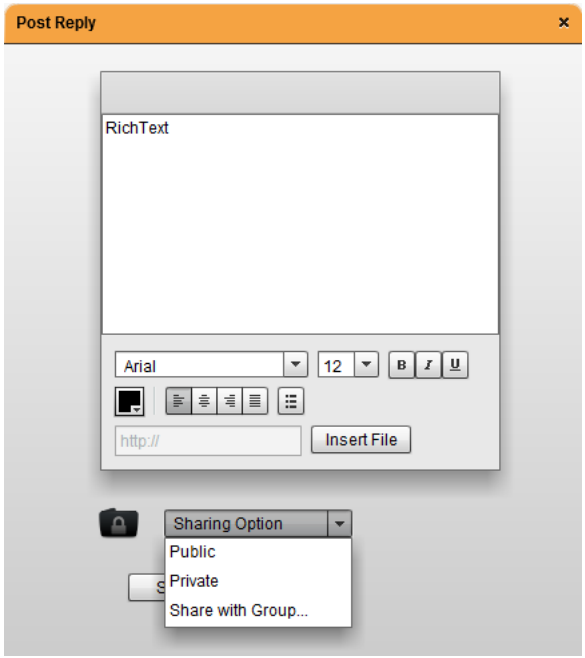

Figure 3.12: Adding an annotation

representation that enables the user to add simple text, rich text, images and other files. Figure 3.12 shows the pop-up window for adding annotations. To add a file to an annotation, users can select the `Insert File` option. This option will pop up a window for uploading files to the server. Once a file has been uploaded, a reference link is inserted in the WYSIWYG text area. If the annotation is requested by other users, these files can be downloaded by clicking the link in the annotation. Figure 3.13 shows the popup window for uploading files. Next to these options, there is also the possibility to record audio and video via a webcam. The flash player will pop up a window for the user to record an audio/video file. Once the file is created the user can upload it as an annotation by inserting it as a file as explained before.

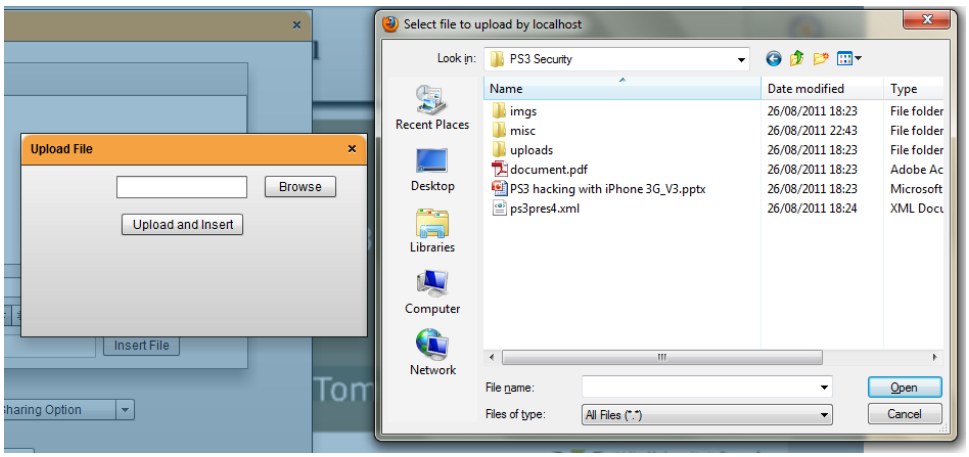

Figure 3.13: Inserting a file as an annotation

The proposed options are unique in lecture capturing tools and they support our vision that students should have full control over their learning experience. Giving students more ways to interact with lectures, professors and other students might result in a better understanding of the content. Therefore, we do not only introduce a more flexible way of annotation, but also the ability to share those annotations.

Sharing Annotations

All the annotations that are made by a user are by default public. However, there are several sharing options. Of the existing solutions that enable the making of annotations, most of them allow making private annotations and only a few incorporate sharing annotations publicly. In this thesis, we introduce various ways of sharing. Not only can annotations be made private or public, they can also be shared with several other users by creating groups and sharing annotations within these group.

Upon adding an annotation, the creator of the annotation thread can set the privacy option. The user can choose between the options: `Public`, `Private` or `Share with Group`, shown in Figure 3.12. By default, annotations are shared publicly with the subscribers of the same course. By setting the sharing option to private, annotations will only be visible for the creators themselves. Furthermore, when the `Share with Group` option is chosen, a pop-up is presented where users can select one of their groups to share the annotation with. Figure 3.14 shows this pop-up, where we can also see that there is an additional option to create a new group. If this option is selected, the creator of the annotation will need to enter a group name and will be able to select users to add to the group. The creator of the group will automatically be appointed as group owner and will be able to manage his groups via the user settings. In Section 3.4 we will elaborate on the user settings.

Only the annotations that are available for the user, will be visualised on the annotation timeline, on the slides and in the annotation list. This includes annotations created by the user, publicly available annotations and annotations that are shared with a group of which the user is a member of.

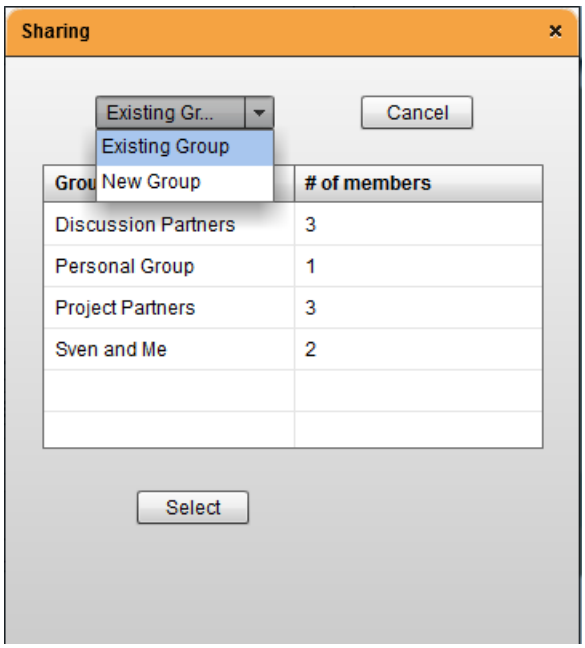

Figure 3.14: Sharing an annotation with a group

3.4 User Settings

Upon accessing the portal, the user has to be authorised to view recorded lectures. Once logged in, the user will be able to navigate to the course section for reviewing recordings. Furthermore, users can personalise their settings via the 'Settings' page shown in Figure 3.15. The main view of the settings shows the personal information of the user, which can be adjusted as well. Additionally, a picture of the user can be uploaded.

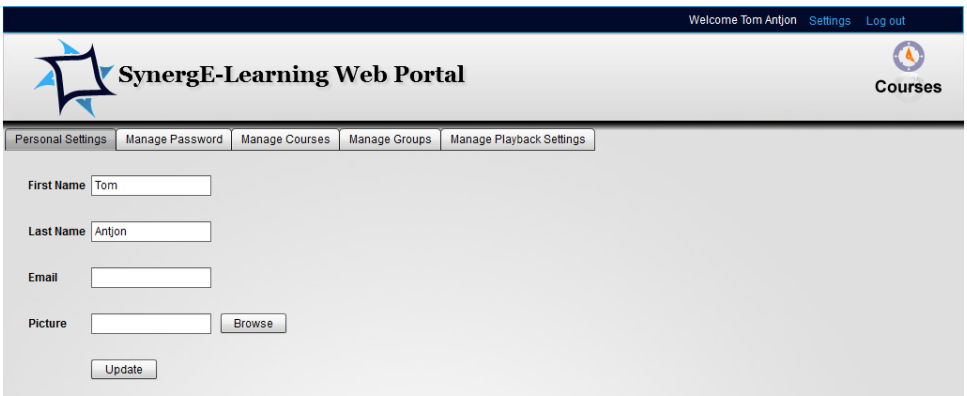

Figure 3.15: The main view of the user settings

The settings page shows a tabbed view of all the settings that can be personalised. Next to changing personal information, users can also change their password, subscribe or unsubscribe to courses, manage their groups or change the settings for reviewing lectures.

For users to be able to review lectures of a certain course, they have to subscribe to that course. This can be done via the `Manage Courses` tab of the user settings shown in Figure 3.16. This page shows all the courses that are available in the web portal. The user can filter the list on name, which makes quick searches possible. If the user selects a course, the description of that course is shown below the list. The user can then proceed with subscribing to the course. Lecturers are provided with some additional options in this screen. This window also enables them to manage, create or delete their courses.

Figure 3.16: Subscribing/unsubscribing to courses

Figure 3.17 shows the added functions for lecturers. When a user is logged in and the account has a lecturer role, then he is able to control his own courses. Not only the course information can be changed here but also new courses can be created, and existing courses can be deleted. Lecturers can

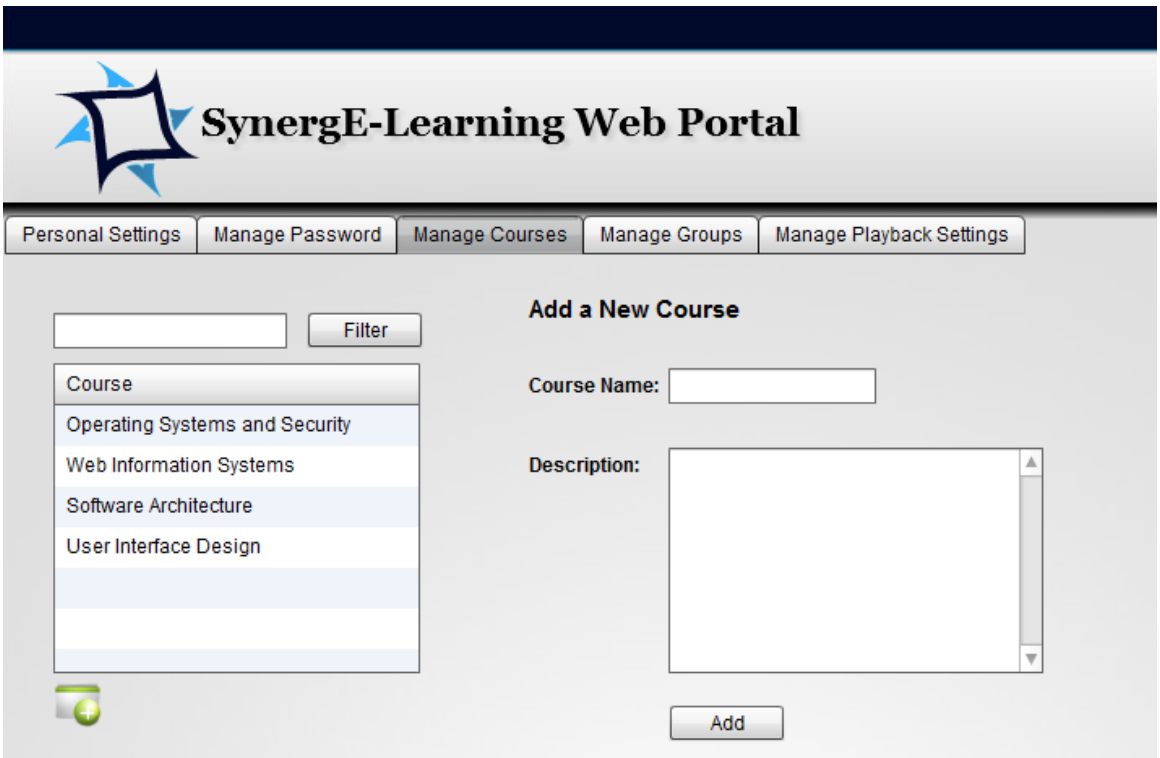

Figure 3.17: Managing courses

also view the users who have subscribed to their courses.

Once users are subscribed to a course, they are able to review the lectures that are created in this course. When the users navigate to the course view, they will be able to select a course to which they are subscribed to. A list of available lectures will be shown next, where users can select one to review. The course view window is shown in Figure 3.18.

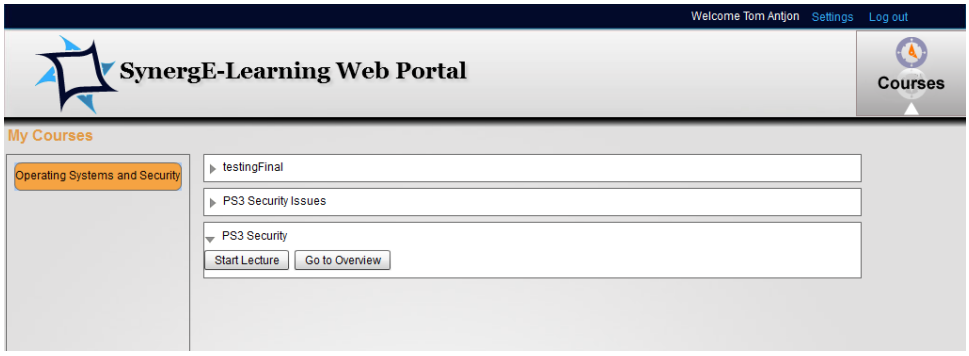

Figure 3.18: Selecting a course and lecture to review

Each lecture contains a link to the actual replay window, but next to that there is also the ability to view and download files that the presenter has attached to the lecture. This can include presentation files, video/audio content or additional files that the presenter has uploaded. Furthermore, the lecture contains a description and a link that goes directly to the slide overview. From this view, the lecture can be replayed directly from a certain slide, by selecting the desired slide.

As mentioned in Section 3.3, users can also manage their groups that they have created when creating annotations. The `Group Management` window is also available via the user settings. In here the users can view the groups that they created or the groups that they are part of and are not the owner of. The options differ based on whether a user is an owner or a regular member. When users

own a group, they can add other users to it or dissolve the group entirely. Upon dissolving a group, the group and all its members is removed. Additionally there is also an option to create a new group. Figures 3.19 and 3.20 show the windows for managing the groups.

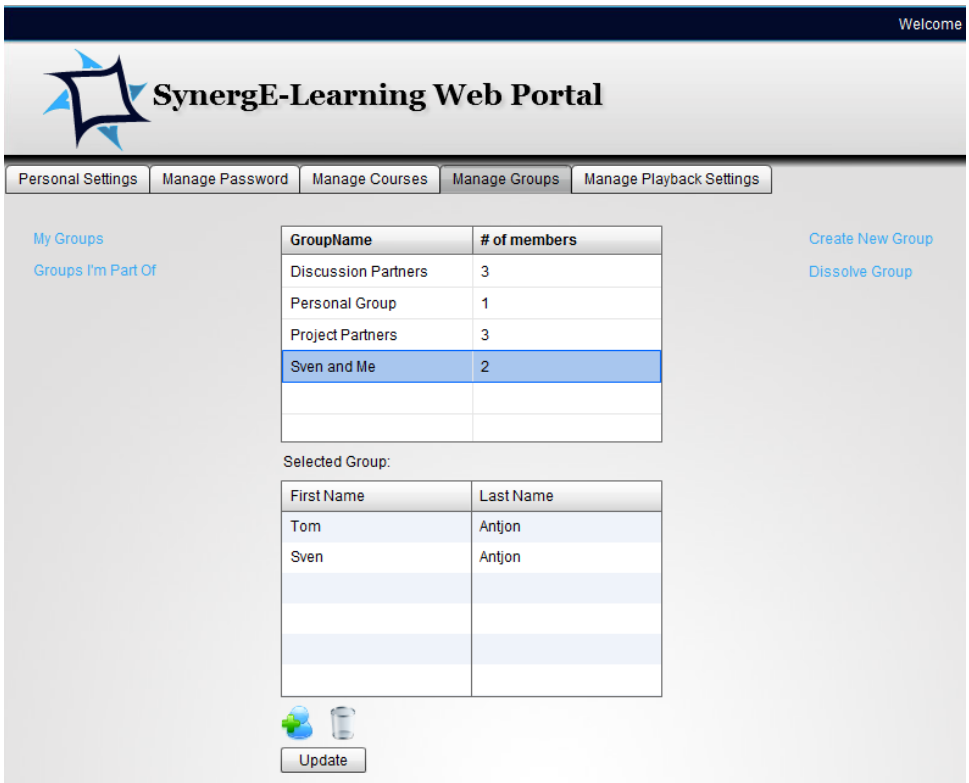

Figure 3.19: Managing the groups of a user

Figure 3.19 shows a list of the groups of a user. Upon selecting a group, an additional list is presented with the users that are linked to this group. The owner of the group can add new members to the group by clicking on the 'Add' button, which will call the 'Add Members' window which is shown in Figure 3.20. Members can also be removed from groups by dragging them from the member list to the recycling bin located below the grid.

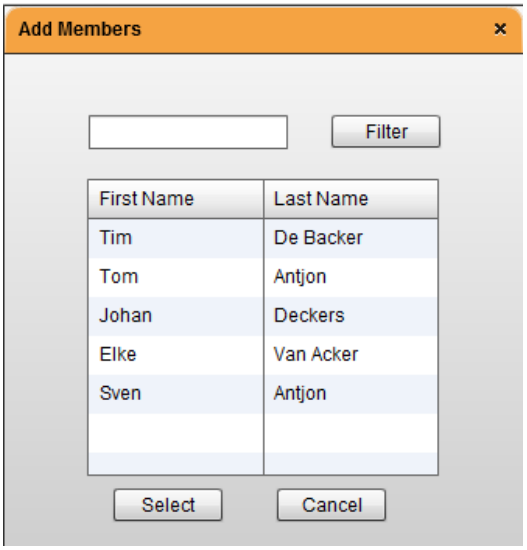

Figure 3.20: Adding members to a group

The group management options are different if a user is just a member of a group and not the owner. Actions are then limited to viewing the other members of the group and leaving a group. Figure 3.21 shows a list with the groups where the user is a member of but is not owner of. Upon selecting a group, an additional list is presented with all the other members of the group and the option to leave the selected group.

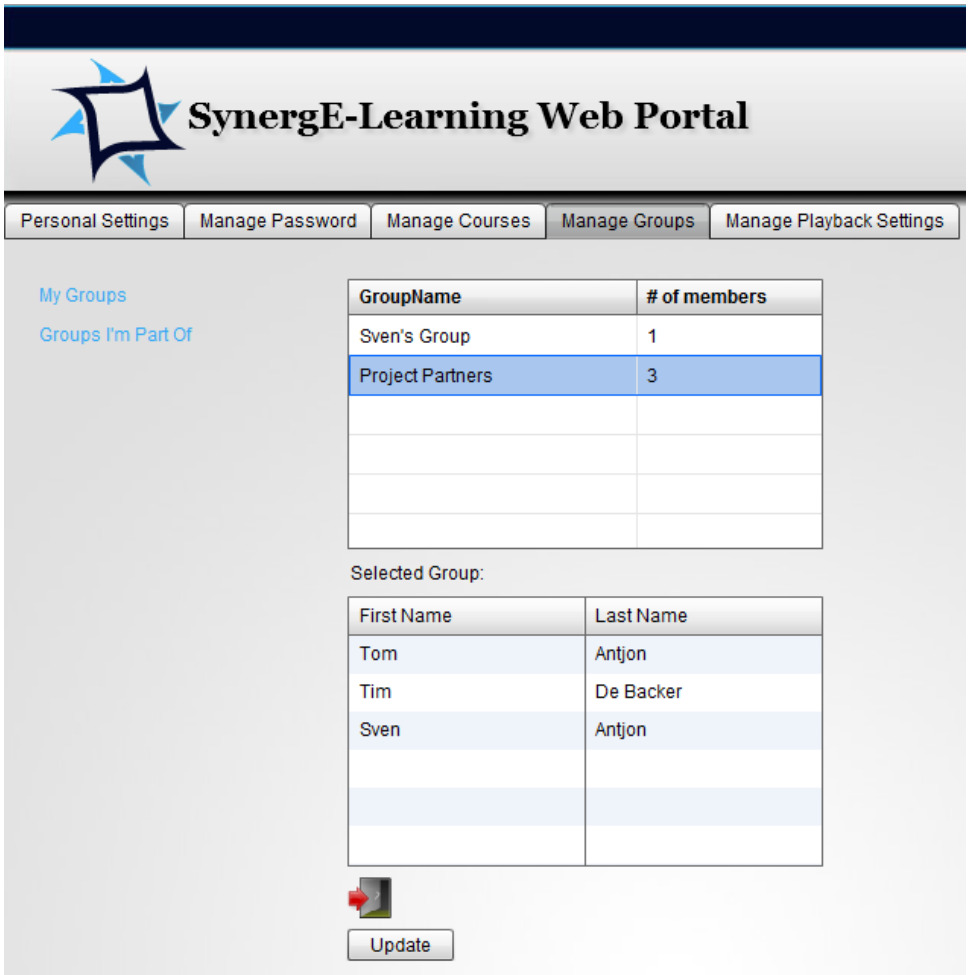

Figure 3.21: Group overview

The last tab of the user settings controls the playback settings. The user can personalise his playback settings for reviewing a lecture. Currently, enabling/disabling on-slide annotations is the only option that can be adjusted. However, new options can easily be defined and linked to user profiles. These settings are shown in Figure 3.22.

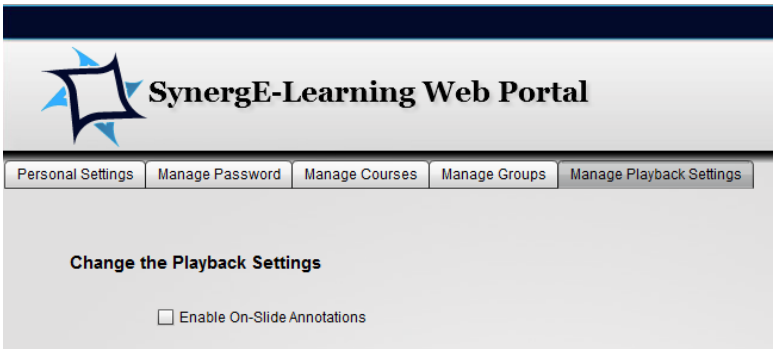

Figure 3.22: Group overview

Chapter 4

Implementation

This chapter will take a closer look to the implementation of the SynergE-Learning Web Portal. We will go through the whole process from creating a presentation with PaperPoint, to reviewing that recorded presentation via the web portal. During this whole process we will elaborate on the used technology and the ideas behind the needed functions. Figure 4.1 shows an overview of the architecture of the SynergE-Learning Web Portal.

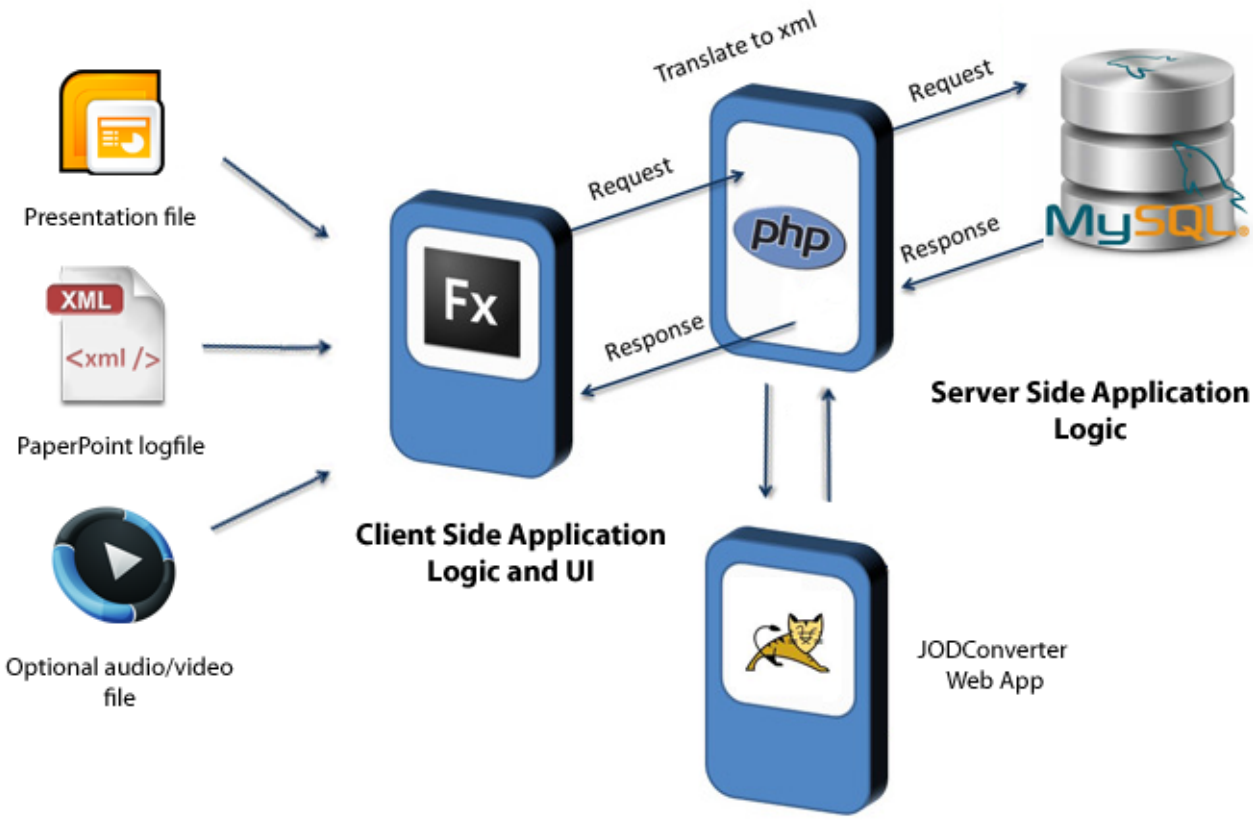

Figure 4.1: An overview of the architecture

4.1 Creating a presentation

In this section we will start by explaining how a presentation is created. This is done via PowerPoint and PaperPoint. However, as explained in Section 2.1.4, PaperPoint is flexible enough to work with other presentation tools. The web portal can also handle other technologies, as long as there is a logfile available which follows a certain template.

4.1.1 PaperPoint

The SynergE-Learning Web Portal currently uses PaperPoint logfiles to recreate a lecture. A PaperPoint logfile is a recording of a PaperPoint session. This file contains all the actions that are undertaken by the presenter with the digital pen during the presentation.

A PaperPoint enhanced presentation does not require lots of extra work. Users can simply create presentations in their favourite presentation tool. Once created, handouts can be created and enhanced with PaperPoint functions as explained in Section 2.1.4. Once the handouts are created, a PaperPoint session can be started. When PaperPoint is loaded, the system asks for the presentation that the user wishes to present. He will also have to ability to create a logfile of the lecture.

This logfile will be needed to recreate the lecture in the SynergE-Learning Web Portal. Together with the presentation and optional audio/video file, the logfile is uploaded when creating a new lecture on the web portal. PaperPoint is currently equipped with a replay function that can re-enact the whole recorded presentation within the software tool of that presentation. For the web portal, the same reasoning has been followed. However, our implementation is web-based and does not require actual presentation software. The only requirement for the web portal is a web browser that is equipped with a Flash Player.

The PaperPoint XML file contains a list of events, that represent pen actions. Currently there are 6 different actions that are recorded and are useful for the replay. However, PaperPoint as well as the web portal are flexible enough to easily extend the logging with newly defined actions. All of the events include a timestamp, the name of the method that is used and a number of arguments for that method. The events that are the most useful are the actions that record the navigation, including the `showSlide`, `showFirstSlide`, `showPreviousSlide`, `showNextSlide` and `showLastSlide` actions. Furthermore, the actions that record the annotations made by the pen are used to recreate the annotations on the slides in the web portal. Figure 4.2 shows an extract of a PaperPoint logfile which recorded the navigation to the 5th slide, followed by drawing an annotation on the slide.

4.1.2 The Web Portal

Now that all the resources are available to recreate the lecture, presenters can go to the SynergE-Learning Web Portal to add a new lecture to their course. With the intention of open source software, this web portal is built based on Flex in combination with a PHP and MySQL back end.

Flex Framework

Flex [1] is an open source framework for building Rich Internet Applications for web, desktop and mobile devices. The Flex framework provides a declarative language, application services, components, and data connectivity that can be enhanced and extended to personal needs. Flex allows developers to

[1] `http://www.adobe.com/products/flex/flex_framework/`

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<log>
  <event>
    <millis>1300827746125</millis>
    <method>showSlide</method>
    <args>
      <int>5</int>
    </args>
  </event>
  <event>
    <millis>1300827746162</millis>
    <method>setPointerType</method>
    <args>
      <int>2</int>
    </args>
  </event>
  <event>
    <millis>1300827746235</millis>
    <method>drawPoints</method>
    <args>
      <list>
            <point2d>
                <double>876.6410314119779</double>
                <double>201.26896551662477</double>
            </point2d>
            <point2d>
                <double>873.3589800321138</double>
                <double>198.95172413772548</double>
            </point2d>
      </list>
      <int>-2147483647</int>
    </args>
  </event>
  <event>
```

Figure 4.2: Extract of a PaperPoint logfile

build web and mobile applications that share a common code base, reducing the time and cost of application creation and longer term maintenance.

In Flex there are two languages that are being used. To define the user interface layout, appearance and behaviour of a Flex application, MXML is used, which is an XML-based language. The client-side application logic is defined in the object-oriented language ActionScript. A Flex application consists of a single SWF file that is compiled from MXML and ActionScript. The server-side application logic can be defined in any language. Flex is equipped with data transport methods including support for XML over HTTP, Action Message Format (AMF), or RTMP using the Adobe LiveCycle Data Services ES2 module or the open source BlazeDS project.

Flexibility

Flex includes a prebuilt class library and application services that help developers to assemble and build applications. They include rich application components and mobile application components. All these components can easily be extended to the needs of the application. Services include data binding, drag-and-drop management, the display system that manages the interface layout, the style

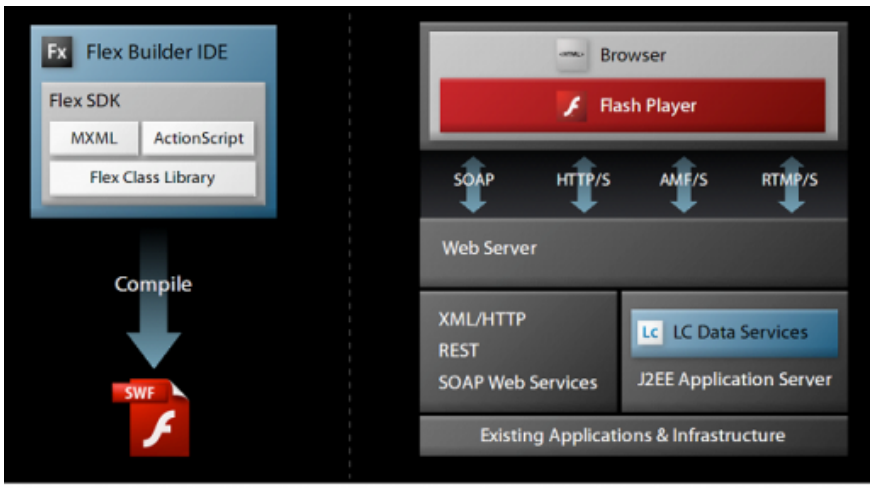

Figure 4.3: Architecture of Flex

system that manages the look and feel of interface components, and the effects and animation system that manages motion and transitions.

Flex applications can easily be ported to mobile devices and to PCs outside the browser because of the Adobe Air runtime. A common base can be created in Flex applications and only a few additional steps need to be taken to optimise the application for Adobe AIR. Afterwards the application can be compiled as a desktop version, or as a mobile application on Android, iOS or RIM.

The freedom of language, the extensibility and the portability are the three reasons why we chose to build the Web Portal on the Flex framework. Flex allowed us to build our own components that can interact with built-in services. These services resulted in a clean user interface with multi-platform compatibility. In the future we would like to port the web application to the desktop and to mobile devices enhancing the flexibility of e-learning for the student.

4.2 Recreating a Lecture

When lecturers wish to add a new lecture, they go to the SynergE-Learning Web Portal and upload the PaperPoint created files. Between uploading and making the recording available for students, there is a whole process of analysing the logfile and creating a new lecture with the corresponding events in the database.

4.2.1 Phase 1: Logfile analysis

In a first phase the logfile is analysed and all the necessary events are extracted from the file and stored in the database. In our implementation, we have chosen a MySQL database. As mentioned before, the data that needs to be stored is rather simple and there is no need for a hi-tech database system. Our choice was also influenced by the fact that we want to provide an open source and free solution.

Process XML

Upon creating a new lecture, it needs to be added to the database first and is linked to a course. A lecture holds information like the title of a lecture, the date, the author and it also holds references to

the logfile, the presentation and additionally the audio/video file, which are uploaded to the server. Once the lecture is created, the system goes on with extracting the events from the uploaded logfile.

Listing 4.1: Process Events Method

```
private function processEvents():void
{
 ...
 // defining variables
 ...
 presStartTime = eventsColl[0].millis;
 presEndTime = eventsColl[eventsColl.length-1].millis;
 tTime = calcTimeDiff2(presStartTime, presEndTime);

 for each(var obj:Object in eventsColl) {
  idx++;
        isRealEvent = true;
        isPen = false;
        if(idx == eventsColl.length)
         isLastEvent = true;

        logEv = new DBLogEvent();
        logEv.setLectureID(lectureID);

        logEv.setStartTime(calcTimeDiff2(presStartTime, obj.millis));
        if(idx < eventsColl.length)
         logEv.setEndTime(calcTimeDiff2(presStartTime, eventsColl[idx].millis));
        else
         logEv.setEndTime(tTime);
        logEv.setTypeIDByMethod(obj.method);
        logEv.setArgs(obj.args)

        switch(obj.method){
         case "showSlide":
          slide = logEv.getArgs().int;
          logEv.setSlideNr(slide);
                break;
                ... // set slide number for other methods and add to collection
                    // unless method == drawPoints
         default:
                if(!isLastEvent)
                 isRealEvent = false;
                break;
        }

        if(isRealEvent)
         addEvent(logEv, isPen);
 }
}
```

Storing Data

Listing 4.1 shows the method that is used to process the events that are retrieved from the logfile. First all the entries from the logfile are stored in a non-typed ArrayCollection via a HTTPService request, but we will elaborate on retrieving data in Section 4.3.1. Now that all the events are stored in an ArrayCollection, we can traverse the collection and filter on the data that we need. For every entry we create a new DBLogEvent object. According to the method of the event, that is retrieved from the XML file, the object receives the correct slide number it refers to and an indication if it is a navigational event or a drawing with the digital pen. As mentioned before, only events that are valuable to us are actually added. These events are navigation events and drawing events. There is one exception, the system also stores the very last event. This event indicates the ending of the presentation.

Each event that is extracted from the logfile receives a unique ID and a reference to the lecture it belongs to. Furthermore, the start time of each event is calculated as well. When the event is of the type 'drawPoints', it means that the event is a draw event and holds an annotation made by the presenter with a digital pen. The explicit content of that annotation gets saved in the database as well. In the future we would like to run OCR software on this data. This way, the content can be made searchable as well. Figure 4.4 shows the model of the database where we can see the implementation of the program on the back-end.

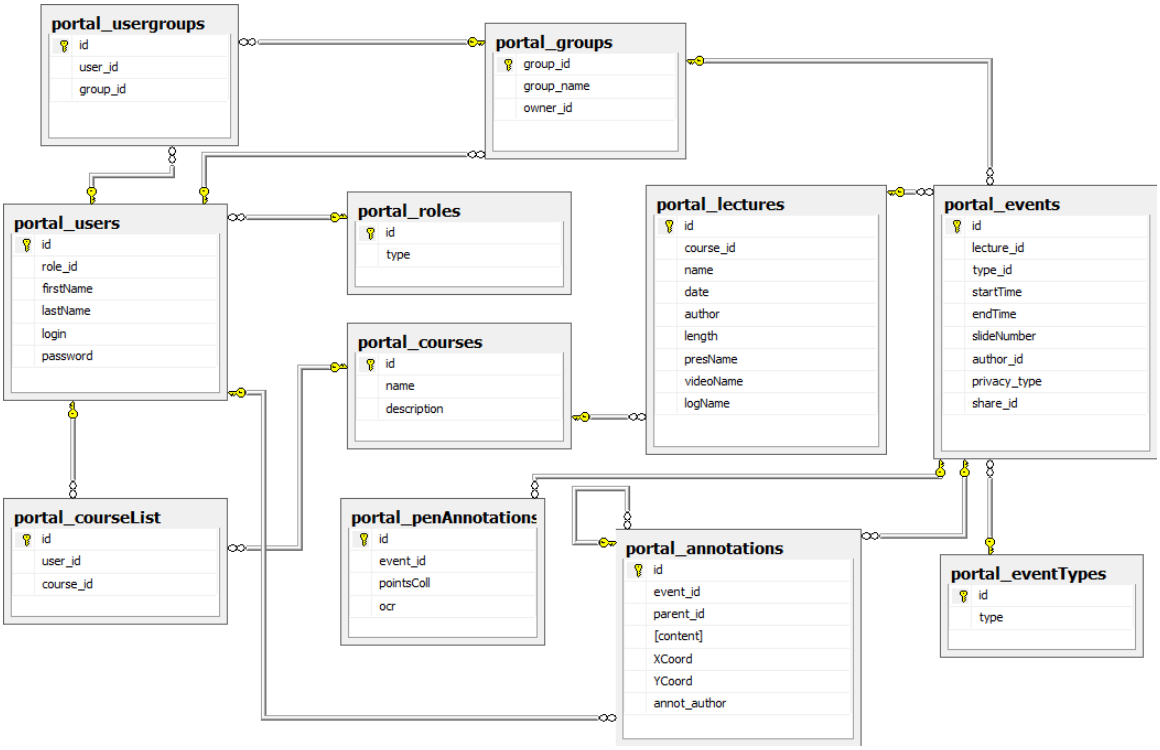

Figure 4.4: The Database Model

Server-Side Application Logic Interaction

As mentioned in Section 4.1.2, Flex only handles the client side of applications. For handling the server-side application logic, a programming language needs to be implemented that will interoperate with Flex. Figure 4.1 shows that in our implementation PHP is in control of the server-side logic. The way that Flex interacts with server-side code can be done on three approaches, depending on the tech-

nology one would like to use. A first approach, which is also our approach, is to use the HTTPService component to send and retrieve data via the HTTP GET and POST operations. The WebService component works with WSDL documents via SOAP-based webservices and the RemoteObject component sends and retrieves data via AMF.

Listing 4.2: Sending event data for server-side processing

```
private function addEvent(ev:DBLogEvent, isPen:Boolean):void
{
 httpService = new HTTPService();
 httpService.url="http://localhost/thesis/php/addEventIF.php";
 httpService.method="POST";
 httpService.addEventListener(ResultEvent.RESULT, updateLatestID);
 if(isPen)
  httpService.send({isPen:isPen,evLecID:ev.getLectureID(),evTypeID:ev.getTypeID(),
evStart:ev.getStartTime(), evEnd:ev.getEndTime(), evSlide:ev.getSlideNr(),
pointColl:ev.getPointCollAsString()});
 else
  httpService.send({isPen:isPen,evLecID:ev.getLectureID(),evTypeID:ev.getTypeID(),
evStart:ev.getStartTime(), evEnd:ev.getEndTime(), evSlide:ev.getSlideNr()});
}
```

Listing 4.2 show the function that is used to send data over to the PHP files for the server-side processing. Here we would like to add a new event, that was extracted from the XML file, to the database. For this a new HTTPService object is created. This object can send data to a specified URL and will pass the answer it receives from that URL to a handler method. In Section 4.3.1 we will go deeper into receiving data and processing it on the client side. Now that the data is sent to the server-side application logic, the PHP files will proceed with processing the data. First, the `POST` variables that are sent are called and a new object is created. Once this is done, a query can be sent to database to actually store the data. To conclude a response is sent back to the client-side application. This response can hold data but also errors and exceptions can be sent and handled in the Flex application.

4.2.2 Phase 2: Presentation Processing

In a second phase, the presentation file must be broken down into separate image files. This way the presentation will be presented in the browser as a slideshow of images based on the events from the XML logfile. This is a unique approach as existing solutions build on capturing the screen during presentation. The consequence of screen capturing is large amounts of data that need high-tech processing equipment. Our approach, on the other hand, is very light and flexible as PaperPoint provides all the presentation information in a small XML file. To process the presentation, we make use of some third-party open source software.

The conversion of a presentation takes two steps. First, the presentation must undergo an extraction process. For this, the application uses JODConvertor[2], this open source technology integrates OpenOffice functionalities for converting documents. The JODConvertor Web Service, which runs on a Tomcat Server, uses OpenOffice.org service to convert the presentation to a PDF document where every slide is presented as one page. In a second step, the PDF that has been created in the first

[2] `http://www.artofsolving.com/opensource/jodconverter/`

step will get processed. Every page of the PDF is converted into a JPEG file that will be used by the application. For this, ImageMagick[3] is used, which is a software suite for creating, editing, composing, or converting bitmap images. In our implementation we use IMagick, a native PHP extension to create and modify images using the ImageMagick API. Figure 4.5 shows the extraction process for the presentations.

Figure 4.5: The Extraction Process

4.3 Reviewing a Lecture

Now every resource is ready and students can proceed by reviewing the newly created lecture. We will go through the whole process it takes for a student to review a lecture.

4.3.1 Accessing Lectures

When students wish to review lectures, they need to have an account on the SynergE-Learning Web Portal. Figure 4.6 shows how students are connected to lectures. Lectures belong to courses where students can subscribe to. This way, courses can be linked to student accounts. Users can also make annotations to the lectures and share them with other users. In Section 4.3.3 we will go deeper into this functionality.

Authentication

The authentication of users in a Flex application is slightly different than on other websites, where normally authentication details are stored in session cookies. As the client side actually is an SWF file and the Flash Player has no session variables, a slightly different approach is used.

In Flex applications, data can be stored on the user's computer for later access by using Local Shared Objects. A Local Shared Object can store up to 100 kB of data without asking permission of the user. Local Shared Objects can store many data types including Number, String, Boolean, XML, Date, Array and Object, unlike cookies that are only capable of storing text values. Custom objects can also be stored by using the `[RemoteObject]` metatag.

The SynergE-Learning Web Portal login component first checks the user's credentials with the database. This is done via a HTTPService request and handled on the server side as explained in

[3] `http://www.imagemagick.org/`

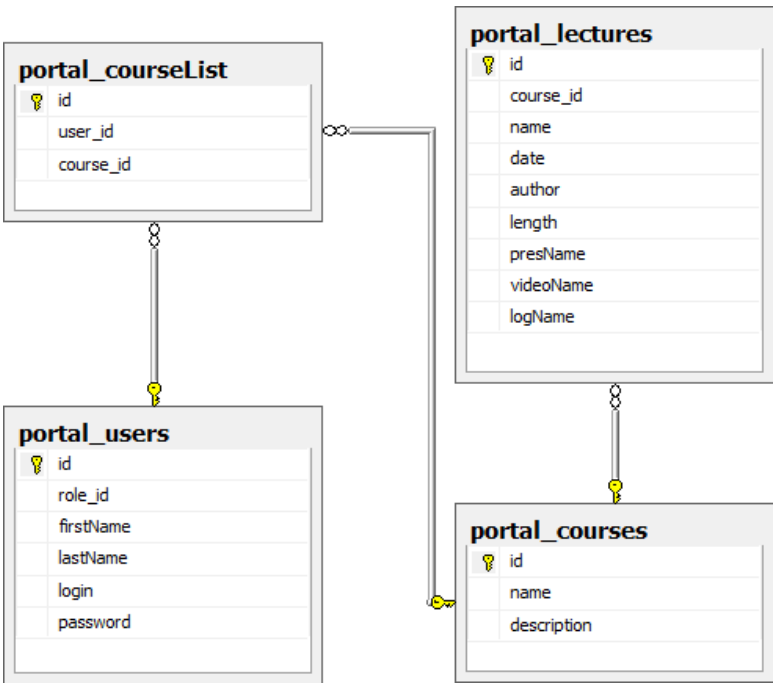

Figure 4.6: Database model of courses and lectures

Section 4.2.1. If the user's credentials are correct, all of the user's data is sent back to the client application. First, we will discuss how this data is sent, followed by how this response is handled on the client side and finally how this data is then stored in a Local Shared Object.

When the server side receives the request to check user credentials a query is sent to the MySQL database. Listing 4.3 shows the PHP functions that take care of handling the MySQL request and response. The checkUser function creates an array which will hold the response of the query. If everything is correct, the array will contain `true` on its first position, indicating that the username has been found in the database. The second position will contain `true`, indicating that the password is correct, and the last position will contain the actual user object from the database. When the username or password do not match the value in the database, the array will be adjusted so that the right error can be deducted and a correct error message can be presented to the user.

Listing 4.3: Checking user credentials

```
function checkUser($login,$pass){
 $checked = array('login'=>false,'pass'=>false,'user'=>null);

 $this->theConn->Open();
 $theQuery = "select * from portal_users as a JOIN portal_roles as b
ON a.role_id = b.id where login = '{$login}'";

 $rs = $this->theConn->ExecuteSQL($theQuery,'B');

 if (is_array($rs)) {
  $checked['login']=true;
        $fetchedUser = $this->getUserFromResult($rs[0]);

        if ($fetchedUser->getPassword() == md5($pass))  {
         $checked['pass']=true;
         $checked['user']= $fetchedUser;
        }
 }
 return $checked;
}

function getUserFromResult($row){
 $user = new user();
  ...
  //creating a user
  ...
 return $user;
}

function getUserCollFromResult($rs){
 $userColl = array();

 foreach($rs as $row) {
  $user = $this->getUserFromResult($row);
        array_push($userColl,$user);
 }
 return $userColl;
}
```

The array containing the response from the database is then passed to a PHP file that serves as a controller between the client side and the server side. This file will create some XML code that can be passed to the client side. Listing 4.4 shows the an extract of the code responsible for creating the XML.

Listing 4.4: Creating the XML response

```
$res = $myUserBH->checkUser($login, $pass);
$return = '<?xml version="1.0" encoding="utf-8"?><data>';

if($res[login] && $res[pass])
{
        $user = $res[user];
        $return .= "<res><code>ok</code></res>";
        $return .= "<user>";
        $return .= "<userID>" . $user->getUserID() . "</userID>\n";
        $return .= "<login>" . $user->getLogin() . "</login>\n";
        $return .= "<pass>" . $user->getPassword() . "</pass>\n";
        $return .= "<firstName>" . $user->getFirstName() . "</firstName>\n";
        $return .= "<lastName>" . $user->getLastName() . "</lastName>\n";
        $return .= "<role>" . $user->getRole() . "</role>\n";
        $return .= "</user>\n";
}
else
{
        $return .= "<res><code>fault</code></res>";
        if(!$res[login])
                $return .= "<fault><message>User not known</message></fault>";
        else
                $return .= "<fault><message>Wrong password</message></fault>";
}
$return .= "</data>";
print ($return);
```

Listing 4.5 shows the function that is responsible for handling the response from the HTTPService Request. The function receives the XML data and can access it via the method `event.result`. In this case, the XML data will contain a root element `data`, a `res` element with a `code` subelement to indicate if the user's credentials were correct. If they were not valid, the function shows a message box with an appropriate error message. If the credentials are valid, the XML code will contain a user data which can be used to create an actual user object. This object will then be stored in a Shared Local Object together with a boolean value indicating the user is authenticated. Upon creating a new Shared Local Object, the `getLocal` function either creates a new object or retrieves the object if it was previously created. Now that a Shared Local Object is available, data can be added to it. To actually write the data into a cookie located on the user's computer, the `flush` method needs to be called. As mentioned earlier, this object has no limitation regarding the data types it can hold. For this, we have made the `User` valueobject class a RemoteClass. Now the user object can be added to the Shared Local Object and stored on the user's PC.

Listing 4.5: Handling the HTTPService response

```
private function resultEventHandler(event:ResultEvent):void
{
 var res:Object = event.result.data.res;
 var data:Object;

 if(res.code == "ok")
 {
  this.currentState="normal";
        data = event.result.data.user;
        user = new User();
        user.UserID = data.userID;
        user.Role = data.role;
        user.FirstName = data.firstName;
        user.LastName = data.lastName;

        var soLogin:SharedObject = SharedObject.getLocal("userData");
        soLogin.data.loggedin = true;
        soLogin.data.user = user;
        soLogin.flush();
 }
 else
 {
  data = event.result.data.fault;
        Alert.show(data.message, "Alert!");
 }
}
```

Now that the user data is stored in a Shared Local Object every component in our component can access it. This can restricts users from accessing pages where they are not allowed and functions to be added according to their user role. Other then user data, data concerning selected courses and lectures are stored as well. This way, components are not bound to sending this data to other components. When components need access to the data of the selected lecture, they can simply call the Flash cookie and retrieve the correct data.

Lecture Overview

Once students are logged in, they can access the posted lectures from the courses that they are subscribed to. When students navigate to their courses, they receive an overview of their subscribed courses. When a course is selected, the view changes to the lectures that are posted under this course and are visualised in a list view. Flex is able to render every item of an ArrayList that is provided to a list. In the most rudimentary form, Flex renders data as plain text, but more complex items can be rendered as well. For this, a specific item renderer needs to be created.

Each lecture in the list will visualise an overview of the data that it holds. This includes the lecture title, date of posting, the author of the posting, a link to download the presentation and a link to view the recorded lecture.

4.3.2 Replaying a Lecture

Once students have selected a lecture to review they are redirected to the `Lecture Viewing` component. On this page, the recorded lecture can be replayed. The slides of the presentations are synchronised with the audio/video file, the annotations are visualised and the page loads will an overview of the recorded lecture. Students can choose a certain slide to start the replaying of the lecture. Figure 4.7 shows the database model of how lectures and events are linked to each other.

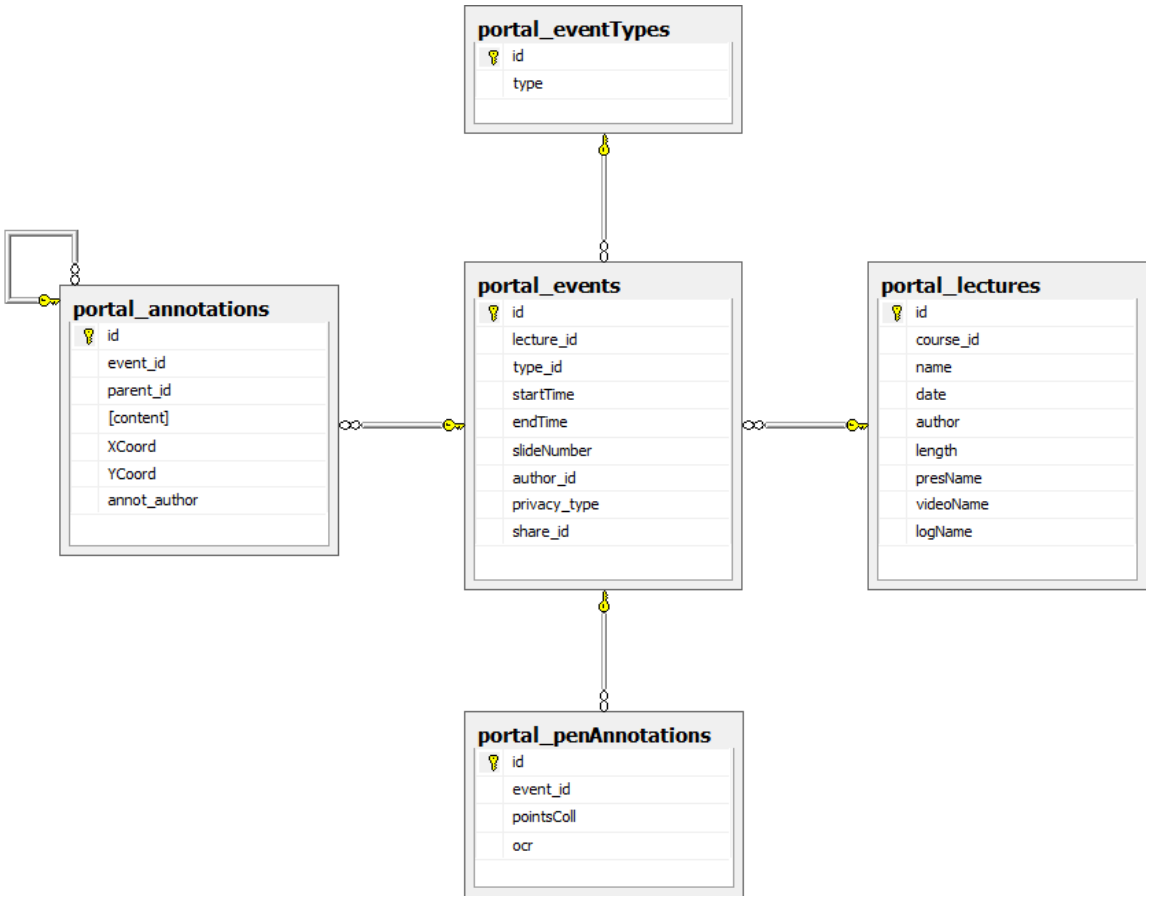

Figure 4.7: Database model of lectures and events

Preparations

The Lecture Viewing component is responsible for setting up the whole replay environment. This includes fetching the lecture data from the database, setting up the video player, creating the slideshow player which processes the events fetched from the database, setting up the annotation overviews and also creating the slide overviews. When a lecture has been selected for replaying, the component queries the database for the lecture data and all the events of the presentation. Once the server side sends an XML list of data from the database, the client side catches the response and converts the XML list into a typed ArrayCollection of DBLogEvents, which are custom objects representing the data extracted from the logfile. Once that data is collected on the client-side, all of the DBLogEvents need to be processed and filtered. All of the navigational events are separated from the annotation events. Listing 4.6 shows the function that processes the events.

Listing 4.6: Analysing the events from the DB

```
private function createEvents(isFirst:Boolean):void{
 slideEventColl = new ArrayCollection();
 drawEventColl = new ArrayCollection();
 slideTickColl = new ArrayCollection();
 var idx:Number = 0;

 for each(var logEv:DBLogEvent in workingEventsColl) {
  switch(logEv.getTypeAsString())        {
          case "showSlide":
           idx++;
                 logEv.setSlideEventNr(idx);
                 slideEventColl.addItem(logEv);
                 if(isFirst)
                  slideTickColl.addItem(logEv.getStartTime() / 1000);
                 break;
          case "showNextSlide":
                  ... // increase index and add to collection
                 break;
          case "showPreviousSlide":
                          ... // increase index and add to collection
                 break;
          case "showFirstSlide":
                  ... // increase index and add to collection
                 break;
          case "drawPoints":
           drawEventColl.addItem(logEv);
                 break;
          case "annotOnSlide":
           drawEventColl.addItem(logEv);
                 break;
          }
         }

         if(isFirst)      {
          createSliderAndOverView();
          totalOfEvents = slideEventColl.length;
          nrOfEvents = slideEventColl.length;
          var evObj:SlideCollEvent =
           new SlideCollEvent('slideCollEvent', slideEventColl);
          dispatchEvent(evObj);
         }
         else
          nrOfEvents = slideEventColl.length;
}
```

When the events are analysed, the type of the event is important. In case that the event is one of the navigational events — `showSlide`, `showNextSlide`, `showPreviousSlide`, `showFirstSlide` — the events are added to a separate ArrayCollection, wrapped in a custom event and dispatched to the

`Overview` components to build the different lecture overviews on which we will elaborate in the next section. If this `createEvents` function is called for the first time, an additional ArrayCollection is created with the timestamps of each slide transition. This collection then used to create the marks on the timeline to indicate these transitions. When the event that is being analysed, is an annotation event, the event will be stored in another ArrayCollection which holds all of the draw events.

Now that all of the events are fetched from the database, analysed and divided in separate ArrayCollections, we can proceed by setting up the slideshow player's media controller. As mentioned previously, the media controller is first enhanced with marks indicating when slide transitions take place. This requires a custom implementation of the Flex HSlider component and a custom skin for it as well. First the HSlider needs to be extended, so that it can hold an ArrayCollection of slide transition events. This collection will hold the data where the marks need to be placed. To actually draw these marks, we need to override the updateDisplayList function that is inherited from the HSlider. This function will re-apply the skin of the custom HSlider, enabling us to draw the ticks after the component is already initialised. Listing 4.7 shows the function in the custom skin that is responsible for drawing the ticks. It is called from the updateDisplayList function.

First, this function calls the hostComponent of the custom skin. This is the custom HSlider that we have created earlier. From this component, we need to request the slide event ArrayCollection. Once we received the collection, we can calculate the correct positions of the marks according to the size of the component. The remaining code of the custom skin takes care of the visual properties of the mark such as size and colour.

Listing 4.7: Drawing transition marks

```
public function drawTicks():void
{
 // dig up to the custom Slider's skin to find information on the
 // arraycollection and size values of the custom slider

 var sliderSkin:SlideShowSliderSkin =
  (this.hostComponent.parentDocument as SlideShowSliderSkin);
 var arr:ArrayCollection = sliderSkin.hostComponent._slideColl;
 var max:Number = sliderSkin.hostComponent.maximum;
 var size:Number = width / max;
 var numberOfTicks:Number = arr.length;

 ticksGroup.removeAllElements();
 ticksGroup.width = 0;

 for (var i:int = 0; i < numberOfTicks; i++)
 {
  var tick:Tick = new Tick();

          tick.y = 0;
          tick.x = (arr.getItemAt(i) as Number) * size;
          ticksGroup.addElement(tick);
 }
}
```

Once the marks are in place, the system must set up a mechanism to actually trigger the transitions. The on-slide annotations must be triggered as well. However, we will come back to annotations in Section 4.3.3. Timer objects are used to trigger all of the events that are previously calculated. Before starting the actual playback, the first timers are set up. For this the first event is retrieved from the collection. The system requests the start time of the event, calculates the time difference and sets up the timer. Pressing the 'Play' button on the media controller, starts the timer and once the timer has ran out, a TimerEvent will be triggered that makes the transition and sets up a new timer for the next transition event. Listing 4.8 shows an extract of code that is used to define the first timer.

Listing 4.8: Setting up the slide timer

```
...
//Set slide timer
var ev:DBLogEvent = slideEventColl.getItemAt(slideEventIndex) as DBLogEvent;
nextSlide = ev.getSlideNr();
endSlideTime = ev.getStartTime();

slideTime = calcTimeDiff2(presStartTime, endSlideTime);

slideTimer = new Timer(slideTime,1);
slideTimer.addEventListener(TimerEvent.TIMER, changeSlide);
...
```

Playback

Once users started the playback function of the lecture, all of the previously defined timers start running. One timer controls the horizontal slider and moves it every second, another timer controls the transitions as mentioned in the previous section, and there are two timers that control the drawing of annotations. If we take a look at Listing 4.9, we can see an example of what happens when a timer runs out of time and the fired event is caught.

The first thing that is done, is displaying the new slide as this event triggers a slide transition. Once the new slide has been loaded, all of the annotations that are made for the previous slide need to be removed. The annotations are not actually drawn on the slides, but on a transparent layer above the slide. This needs to be wiped clean for every slide transition. Now that everything is in order for the current slide, a new slide transition timer needs to be created to trigger the next transition. As long as there are transitions, new timers get created when handling a timer event.

Listing 4.9: Setting up the slide timer

```
private function changeSlide(e:TimerEvent):void
{
 imgSlide.load(imgPath+pictures.getItemAt(nextSlide - 1));
 imgAnnotation.graphics.clear();

 while(imgAnnotation.numChildren > 0)
 {
  imgAnnotation.removeChildAt(0);
 }

 slideEventIndex++;

 eventIdx = (totalOfEvents - nrOfEvents) + slideEventIndex;
 var evObj:SlideChangeEvent = new SlideChangeEvent('slideChangeEvent', eventIdx);
 dispatchEvent(evObj);

 if(slideEventIndex < slideEventColl.length)
 {
  var ev:DBLogEvent = slideEventColl.getItemAt(slideEventIndex) as DBLogEvent;
        nextSlide = ev.getSlideNr();

        var newSlideTime:Number =
         calcTimeDiff2(presStartTime + (timeStamp * 1000), ev.getStartTime());
        slideTimer = new Timer(newSlideTime,1);
        slideTimer.addEventListener(TimerEvent.TIMER,changeSlide);
        slideTimer.start();
 }
}
```

Where the lecture has been started and plays continuously until the end, the timers just need to pop the top event from the ArrayCollections. However, when the user interacts with the media controller and skips to a certain timestamp, the ArrayCollections need to be recalculated. In Flex there is no implementation of collections with a hash technique. Therefore, when the ArrayCollections need to be recalculated, they need to be traversed manually to look for the position the newly selected slide. After that, a new ArrayCollection needs to be created that only consists of the events that take place after the current event. However, the annotation events that took place between the current slide's start time and the current timestamp should be added as well. To summarise, when manually skipping the lecture to a certain timestamp, first the ArrayCollection needs to be adapted to only contain the remaining events and after that the ArrayCollection should be extended to also hold the annotation events that should already be present at the new slide. In case the user skips forward, the new ArrayCollection should be reduced and if the user skips backwards, it should be enlarged.

Slide Overviews

Existing lecture capturing tools do not pay a lot of attention to overviews of the recorded lectures. If we have a look at the comparison table of the existing solutions in Section 2.3.4, we can see that most of the solutions do provide an overview. All of discussed solutions have a linear overview of the lecture. However, this does not give a clean overview as most presentation are not linear at all. There

is no indication how many times a slide has been used, nor does it represent the path that has been traversed.

One of the most important points of this thesis was to provide flexibility for the student to choose an overview that suits their needs. Some students may find it useful to have a linear overview. We also provide more intuitive forms of overviews of a presented lecture. Next to the linear overview, we constructed two other overviews. The first implementation, introduced in Section 3.2, shows the path that a presentation has been through, visualised by a line. On this line the slides are represented. The slides that have been visited multiple times are placed under each other. The line shows a much more accurate view of how the presentation was given. The second overview visualises the slides as an enhanced linear overview. It is enhanced in the sense that the slides are numbered and multiple instances are shown below each other. Each slide is also shown with his timestamp for a better visualisation when the slides are visited during the presentation. To change the detail level of the overview zooming tools have been implemented as well.

To visualise these overviews, an ArrayCollection of all the slide transition events, previously calculated in the slideshow player component, is passed to the overview components. Each component processes the events and creates his own collection of 'Slide' objects. During this process, the components define the properties of the slides, which are different for every implementation. The most noticeable difference lies in the calculating of the positioning of the slides to create the right effect for the overview. Listing 4.10 shows the process function for the path overview.

Listing 4.10: Processing events for a path overview

```
public function createOverview(coll:ArrayCollection, path:String):void
{
 ...//defining variables
 for each(var ev:DBLogEvent in coll)
 {
  slide = new Slide();
        slide.setEventNr(ev.getSlideEventNr());
        slide.setImagePath(path + "slide-" + (ev.getSlideNr()-1) + ".jpeg");
        slide.setSlideNr(ev.getSlideNr());
        slide.setEvent(ev);

        if(ev.getSlideNr() < highestSlideNr)
         rowNr++;

        highestSlideNr = ev.getSlideNr();
        slide.setCol(highestSlideNr - 1);
        slide.setRow(rowNr);
        slide.setXVal((highestSlideNr - 1) * imgWidth);
        slide.setYVal(rowNr * imgHeight);

        overviewColl.addItem(slide);
 }

 pathOverview.buildOverview(overviewColl,true);
 createSingleSlideOverview(coll,path);
}
```

Now that the `Slide` objects are created from the events, they can be sent to the visualisation component that will build the actual overview. Listing 4.11 shows the code that is used to build the actual overview. Less important parts of the code were omitted. This function takes an ArrayCollection of `Slide` objects, traverses the collection and creates a new `SlideComponent`. This component consists of the slide images belonging to the slide event and is represented in a bordercontainer. Depending on the second parameter the components will be connected with a line, revealing the presentation path. If this parameter contains the boolean value `true`, the components are first extended with a starting and ending point for the path to be connected to. This is especially useful to connect two components who are not adjacent, but are separated by more than one column or are positioned on different rows. Now that the component is ready, it can be connected with the previously added component by making use of the draw function, which can be seen in Listing 4.13 and is explained in Section 4.3.3.

Listing 4.11: Processing events for a path overview

```
public function buildOverview(slideColl:ArrayCollection, doDraw:Boolean):void{
 ... //defining variables
 for each(var slide:Slide in slideColl) {
  idx++;
  var comp:SlideComp = new SlideComp();
  comp.addEventListener(MouseEvent.CLICK, pushSelectSlideEvent)
  comp.slideData = slide;

  if(slide.getRow() > currentRow) {
   currentRow = slide.getRow();
   currentCol = slide.getCol();

   if(doDraw) {
    ...//draw connection line between previous component
    //and new component
   }
  }
  else {
   if(slide.getCol() > currentCol) {
    var colDiff:Number = slide.getCol() - currentCol;
    currentCol = slide.getCol();
    if(doDraw) {
     ...//draw connection line between previous component
              //and new component if not adjacent
    }
   }
  }

  comp.x = slide.getXVal();
  comp.y = slide.getYVal();

  compColl.addItem(comp);
  this.addElement(comp);
 }
}
```

Switching from one overview to another can be done quickly and easily as the overview components are based on the `Navigationcontent` component. This way, switching between components can be done by using the `Tabbar` component without reinitialising the components every time they are accessed. Once one of the overviews has been selected, users receive a full overview of the recorded lecture, showing thumbnails of the slides accompanied with the timestamp of the transitions. Users can use the overview to skip the lecture to a certain part. When users click on a thumbnail of a slide, the slideshow player sets the recording to the corresponding timestamp and resumes the replaying from that part. To position the slideshow player correctly, the overview component needs to notify the player. This can be done by dispatching a custom event from the overview and catching this event with the slideshow player.

Listing 4.12: Custom event

```
package events
{
 import flash.events.Event;
 import valueObjects.Slide;

 public class SlideSelectEvent extends Event
 {
  public var slide:Slide = new Slide();

        public function SlideSelectEvent(type:String, slide:Slide)
        {
         super(type);
         this.slide = slide;
        }

        override public function clone():Event
        {
         return new SlideSelectEvent(type, slide);
        }
 }
}
```

Listing 4.12 shows how an example of a custom event. This `SlideSelectEvent` takes a `Slide` valueobject as parameter and sets it as a property. The clone function also has its own implementation in this custom event. This function enables the possibility to bubble this custom event. Bubbling events is used when events need to traverse several layers of components. When bubbling is disabled, only the parent of the component can register an eventListener to catch events. However, when bubbling is enabled, events can be passed along to higher level components until a component, with an eventListener registered for this type of event, catches it. To re-dispatch events like this, the clone function must be implemented. In our implementation the overview components use the `dispatchEvent` function to pass the custom event. To make use of this extended event, we need to register it within the classes metadata compiler directive, otherwise the event will not be recognised. Once this is done, the `viewLecture` component can register a listener on this newly defined custom event. When one of the overviews dispatches the `SlideSelectEvent`, the `viewLecture` component catches the event, extracts the slide information and fires the function `goToEvent` in the slideshow player which positions the lecture to the chosen slide.

4.3.3 Annotation Services

The second focal point of the SynergE-Learning Web Portal lies in its annotation service. There are few solutions who support post-presentation annotation services, and if they do it is in a very basic form. This thesis introduces annotation services during presentation and also after a presentation has taken place. In Section 3.3 we presented our methods of visualising annotations of a lecture. As mentioned before, there are two types of annotations, annotations made by a digital pen and annotations made via the Web Portal. Both annotation types have a different way of visualisation as well. In this section we will first take a look on how the visualisation is done, afterwards we will focus on how annotations are represented.

Drawing an Annotation

The first type of annotations, the annotations made with a digital pen, are stored in the database as a collection of points and are linked to an event of the type `drawEvent`. When the lecture is being played back, there is a timer that triggers events when the timestamp of the playback is equal to the timestamp of an event that belongs to this lecture. If an event happens to be a `drawEvent`, the program will draw the collection of points that accompanies the event onto the current slide, creating an annotation.

Listing 4.13: DrawPoints Method

```
private function drawPoints(pointsColl:ArrayCollection):void
{
 var nrOfPoints:Number = pointsColl.length;
 var draw_commands:Vector.<int> = new Vector.<int>(nrOfPoints, true);
 var draw_coord:Vector.<Number> = new Vector.<Number>(nrOfPoints * 2, true);

 for(var i:Number = 0; i < nrOfPoints;i++)
 {
  if(i == 0)
   draw_commands[i] = 1;
  else
   draw_commands[i] = 2;
  draw_coord[i*2] = pointsColl.getItemAt(i).double[0] * scalingsFactorX;
  draw_coord[(i*2) + 1] = pointsColl.getItemAt(i).double[1] * scalingsFactorY;
 }

 var lineThickness:Number = 2;
 var lineColor:Number = 0x000000;
 var lineAlpha:Number = 1;

 imgAnnotation.graphics.lineStyle(lineThickness,lineColor,lineAlpha);
 imgAnnotation.graphics.drawPath(draw_commands, draw_coord);
}
```

In Listing 4.13, we can see the method that takes care of drawing the points onto the slides. The `drawPath` method, a built-in method in ActionScript for drawing graphics, needs two vectors as arguments. In the first vector, the draw commands are stored. There are two commands, the number 1 refers to the command `moveTo`, which makes the graphics drawing tool move to a set of coordinates

that is stored in the second vector. The other command is `lineTo`, this command makes the graphics drawing tool draw a path between the coordinates in the coordinates vector. We can see that this method fills the command vector with `lineTo` commands, except for the first command, which is a `moveTo` command to position the drawing tool. The coordinates vector gets filled with the X and Y values of the points from the point collection. Figure 4.8 shows a few records of the pen annotation table in the database. Next to the collection of points that represent the drawing, these records also hold a reference to the event they belong to.

id	event_id	pointColl
1	6	<point2d><double>820.0256451093232</double><double>433.15862068636665</double></point2d><point2d><double>824.3743631811221</double><double>433.8206896535281</double></point2d>
2	5	<point2d><double>824.3743631811221</double><double>433.8206896535281</double></point2d><point2d><double>831.0205171960031</double><double>434.06896551621367</double></point2d>
3	7	<point2d><double>831.0205171960031</double><double>434.06896551621367</double></point2d><point2d><double>840.374363609053</double><double>434.31724137889927</double></point2d>
4	9	<point2d><double>840.374363609053</double><double>434.31724137889927</double></point2d><point2d><double>846.1487227526485</double><double>434.2034482751684</double></point2d><point2d><double>852.6000049957861</double><double>433.9034482744233</double></point2d>
5	10	<point2d><double>852.6000049957861</double><double>433.9034482744233</double></point2d><point2d><double>859.6461590527468</double><double>433.4896551700759</double></point2d><point2d><double>867.1230823582052</double><double>433.0758620664991</double></point2d>
6	11	<point2d><double>867.1230823582052</double><double>433.0758620664991</double></point2d><point2d><double>872.2528071156859</double><double>432.8551724120727</double></point2d><point2d><double>877.4189991121694</double><double>432.6896551718049</double></point2d><point2d><double>882.5487238696497</double><double>432.5793103452387</double></point2d>
7	12	<point2d><double>882.5487238696497</double><double>432.5793103452387</double></point2d><point2d><double>890.0666728169491</double><double>432.5275862079499</double></point2d><point2d><double>897.4000063667949</double><double>432.6620689661339</double></point2d>
8	17	<point2d><double>935.8000075046832</double><double>435.5586206871887</double></point2d><point2d><double>943.5128282441032</double><double>436.7172413786936</double></point2d>
9	15	<point2d><double>943.5128282441032</double><double>436.7172413786936</double></point2d><point2d><double>947.6153924852356</double><double>436.1379310344827</double></point2d>
10	13	<point2d><double>911.676929865943</double><double>433.4896551709751</double></point2d><point2d><double>918.5794941897068</double><double>433.98620689531856</double></point2d><point2d><double>925.0512892649724</double><double>434.4827586196619</double></point2d>

Figure 4.8: An extract of the database showing the pen annotations

Post-Lecture Annotations

The second type of annotation are the post-lecture annotations, which are annotations made via the SynergE-Learning Web Portal. These annotations can be added in various ways and will be visualised in different ways as well. A first visualisation method is the on-slide visualisation. The on-slide annotations can be added by selecting a position on a slide. A new annotation event will be created and will hold a timestamp, a slide number, a location on the slide and the annotation content. This content can be rich text, a link to a document or a link to an audio or video file. When the playback timer triggers the on-slide annotation event, the annotation gets added to the slide at the position that was previously defined. The on-slide annotations are visualised as text bubbles and shows the content of the annotation in a datacontainer. Each of the text bubbles can be dismissed from the slide as well.

We also introduce an overview of the annotations that are made, this way a user can easily navigate to annotations, participate in a discussion or annotate an annotation. We have implemented two overviews, again to provide the student with flexibility. A first implementation as mentioned in Section 3.3, is the annotation bar shown in Figure 4.9. This bar is an extension of the video controller and visualises the annotations by marking the timeline with the annotation timestamps. Clicking on a time indication on the annotation bar, allows the user to add a new annotation on that timestamp. Clicking on an existing annotation, shows the annotation in a pop-up and allows users to comment or annotate the annotation.

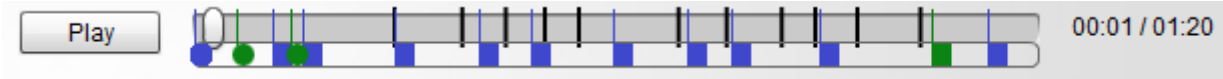

Figure 4.9: The annotation bar which marks the annotations.

The annotation bar is a separate component, which holds annotation components. When the annotation bar is loaded, it requests the annotations from the database via a HTTPService request. Listing 4.14 shows how the response is handled once the data from the database has been stored into a typed ArrayCollection. First, each one of the annotations is evaluated. If the annotation is linked to an event, an annotation component is created and added to the annotation bar. If the annotation is linked to an other annotation, it is added to the component of the parent annotation.

Listing 4.14: Visualising annotations

```
private function visualizeAnnotations():void {
 for each(var annot:Annotation in annotColl) {
        if(annot.getParentID() == 0)
   addToTimeLine(annot);
        else
         addToAnnotation(annot);
 }
}

private function addToAnnotation(annot:Annotation):void {
 var annotComp:AnnotComponent;
 var i:Number = 0;

 for each(var comp:AnnotComponent in compColl) {
  if(comp._id == annot.getID()) {
         annotComp = comp;
         break;
        }
 }

 annotComp.addAnnotation(annot);
}

public function addToTimeLine(annot:Annotation):void {
 var timeLineWidth:Number = this.width;
 var milWidth:Number = timeLineWidth / tTime;

 var annotComp:AnnotComponent = new AnnotComponent();
 annotComp.setType(annot.getShareType());
 annotComp.name = annot.getID().toString();
 annotComp._id = annot.getID();
 annotComp.addAnnotation(annot);
 addEventListener("annotCompClicked", annotCompClickedHandler);
 annotComp.xval = (annot.getStartTime() / 1000) * milWidth;
 compColl.addItem(annotComp);
 this.addElement(annotComp);
}
```

Visualising Annotations

Section 3.3 introduced the annotation overviews that we provide. We have the annotation bar to visualise the on-slide and timestamp annotations. And next to the annotation bar, we also have a grid view of all the annotations. This overview shows a list of the annotations where only the parent annotations are shown. Annotations that are linked to other annotations can be viewed when selecting the parent annotation. The annotation table shows the timestamp of the annotation, a teaser of the annotation and the number of annotations that are linked to it. Again, this overview is a separate component and builds its content from evaluating the annotations.

When an event is selected, whether this is done in the annotation list or by selecting an annotation from the annotation bar, the annotation is visualised. This visualisation contains a list of annotations shown in a threaded form. Each annotation is shown in the list and can be annotated as well. For this, the window is equipped with the option to switch the screen to input-mode where annotations can be added via a WYSIWYG text area. The content of the annotations is rich text, and can contain links to other files. When clicking these links, the files can be downloaded.

Adding an Annotation

Section 3.3 showed how an annotation can be added. The component responsible for the adding annotations pops up when the annotation bar is clicked or when a position is selected on the presented slide. To handle pop-ups Flex has uses a `PopUpManager`. This manager controls all the pop-ups, it creates them, passes events, maintains them and removes them. The components that are used to pop up are usually implementations of a `TitleWindow`. However, there is also the possibility to create custom components that can pop up.

When adding an annotation, a new event will be created on the timestamp and will receive the location coordinates in case of an on-slide annotation. The WYSIWYG window can be used to add rich text content to the annotation. There are also options for the user to upload files and insert links to those documents as annotations.

Listing 4.15 shows the code that creates the component for adding annotations and presents it as a pop-up. The component is created and is provided with an `eventListener` that will handle adding a new annotation to the annotation overviews. The handler for this event will be triggered once a new annotation is created in the pop-up window and has been sent to the database. Furthermore, the code checks if the function call has been made for an on-slide or an off-slide annotation. In case of an off-slide annotation, the annotation bar triggered the function because a user has clicked on the bar. The annotation bar has no knowledge of what timestamp is associated with the click event, only the x and y value of where the click event took place are known. However, the timestamp can easily be calculated with this information. First, we need to what time unit matches one pixel, now we can calculate the timestamp by multiplying this time unit with the x value of the mouse click. In case of an on-slide annotation, the timestamp is already known but we need to add additional data on the position of the annotation. Once everything has been set, all the information for making an annotation will be passed to the component so it can proceed with adding the content to the annotation and committing it to the database.

Listing 4.15: Method to control the 'Add Annotation' component

```
private function showAddAnnotation(type:String, xval:Number, yval:Number = 0):void
{
 var popAnnot:AnnotThread =
  PopUpManager.createPopUp(this, AnnotThread, false) as AnnotThread;
 var milli:Number;

 popAnnot.title="Add an annotation";
 popAnnot.setStyle("borderAlpha", 0.9);
 popAnnot.addEventListener("newAnnotEvent",captureAnnotEvent);

 if(type == "annotBar")
 {
  milli = calcMilliFromXVal(xval);
 }
 else
 {
  milli = timeStamp;
  popAnnot.setCoords(xval,yval);
 }

 popAnnot.newAnnotation(lecID,milli,type);
}
```

Users can add rich content to annotations via a WYSIWYG text area. Flex offers a standard component, the `RichTextEditor` which can handle the input of rich text. However, the standard functions did not meet all of our demands. Therefore, the component is extended so that it can also handle adding files as content. Upon initialisation of the component, an extra button is added to upload files. When that button is clicked, a new window opens to upload files to the server via a PHP program. Once a file has been uploaded, a reference link is returned that can be added to the `RichTextEditor`. Next to adding files, rich text can be used to layout the content of an annotation. The current implementation supports customisable fonts, aligning and lists.

4.3.4 Sharing Annotations

While adding new annotations, users can also define privacy settings. By default, annotations are shared publicly. However, students can define their own groups for sharing annotations. Each annotation event receives a privacy number which indicates whether the annotation is public, private or shared with a group. In the latter case, the annotation event also holds a sharing ID. This ID is a reference to the ID of the group with whom the annotations is shared with.

Figure 4.10 shows how sharing annotations is implemented in the database. As explained earlier, annotations are linked to events that belong to a lecture. When an annotation is shared with a group of users, the event has a share ID that points to the group it is shared with. A group contains several people, therefore, the `portal-usergroup` table has been implemented as a reference table. Groups, as well as events and annotations, also have an owner, which is a foreign key to the user table.

Whenever users wish to view a lecture, only the annotations that they are allowed to see should be requested from the database. This means, users should only have access to the publicly available annotations, private annotations that they made themselves and annotations that are shared with a

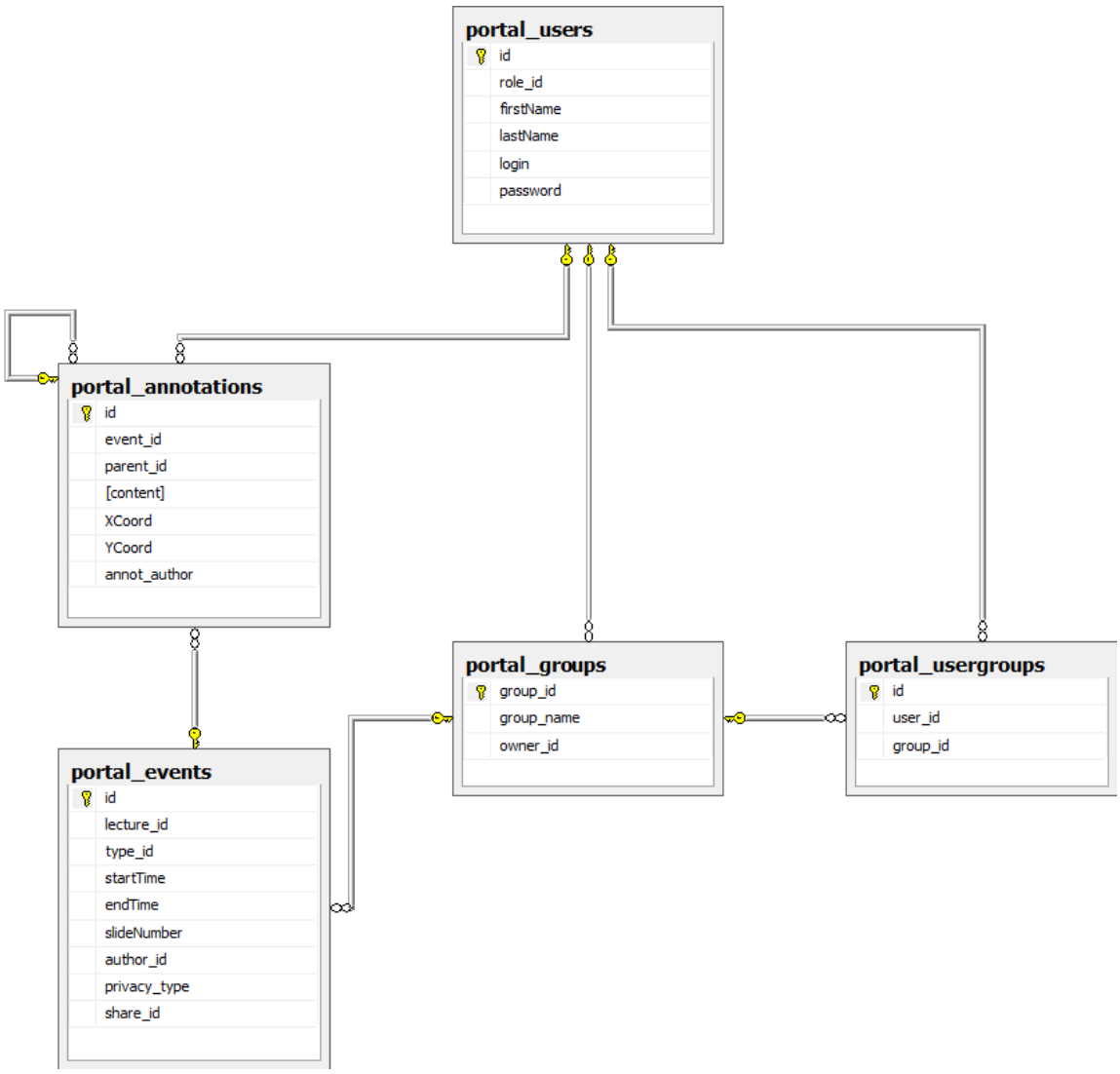

Figure 4.10: DB implementation of sharing annotations

group that they are part of. Listing 4.16 shows how this is translated into a query.

Listing 4.16: Querying the database for annotations

```
function getAnnotationsByLecId($lecID, $userID)
{
 $this->theConn->Open();
 $theQuery = "SELECT * FROM portal_events as a JOIN portal_annotation as b
 ON a.id = b.event_id WHERE a.lecture_id = '{$lecID}' AND
 ((a.share_type = 0) OR (a.share_type = 1 AND a.author_id = '{$userID}')
 OR (a.share_type = 2 AND a.share_id
 IN (SELECT group_id FROM portal_usergroups WHERE user_id = '{$userID}')))";
 $rs = $this->theConn->ExecuteSQL($theQuery,'B');
 return $this->getAnnotationCollFromResult($rs);
}
```

Chapter 5

Discussion

In this thesis we have introduced the SynergE-Learning Web Portal, a web portal that enables replaying lectures recorded with the PaperPoint technology. There are already several lecture capturing and replaying tools available on the market. However, the existing solutions do show some shortcomings. Based on our investigations, we have highlighted some of the shortcomings and attempted to overcome those in our implementation. In this chapter we will discuss the issues that we have mentioned in Chapter 1 and compare our solution to the existing lecture capturing tools. The following list shows a summary of the issues:

- Provide various presentations overviews: linear overview, path visualisation or single-instance slide overview;
- Provide various methods to annotate: annotate a time event, annotate a slide section or annotate an annotation;
- Provide different annotation content: text, image or audio/video;
- Provide flexible sharing methods for annotations: private, share with person, share with group or share publicly;
- Provide various annotation visualisation methods: list view, timeline visualisation or on-slide visualisation;

We will end this chapter with a adjusted comparison of the existing tools.

Presentation Overview

When we introduced different lecturing capturing tools in Section 2.3, we have seen that most solutions only pay little attention to providing a decent overview of the recorded lectures. Elluminate does not even provide any overview whatsoever. The other discussed lecture capturing tools analyse recorded lecture during the upload and derive a linear overview from the analysis. However, most presentations are not presented in a linear fashion. Often it occurs that presenters go back and forth between slides. A linear overview does not visualise this effect clearly. It does not provide the reviewer with any information on how many times a certain slide has been presented. The reviewers themselves must notice that a slide has been visited multiple times. This thesis introduced two alternative overviews to visualise the non-linearity.

The first alternative is an enhanced linear overview. In this overview, the linearity is kept intact. However, multiple visitations of slides are shown below each other accompanied with their timestamp.

This way, each slide instance is represented in one column, and each row indicates a visitation of that slide.

The second alternative overview shows the presentation path. This path is represented as a directed acyclic graph, where the nodes are the visited slides and the directed edges indicate the order of the visitation. This path visualises the storyline of the presentation. Reviewers can quickly see the order of the presentations, and the zooming tools that are provided with the overview can visualise a bigger picture. Linear overviews are not capable of showing more information than what the previous slide was or what the next slide will be. The overviews that we introduce in this thesis are able to give a clearer overview and reveal more information by doing so.

Annotation Methods

If we take a look to other lecture capturing software, only 50 percent support post-lecture annotating. The software that does support annotating, only allows it on a very low level. All of the annotations are linked to a certain time event of the presentation, no other indication of what is precisely annotated exists and there is no possibility to annotate other annotations.

Our goal in this thesis was to provide students with flexibility. Several methods for annotating have been introduced so the students can clearly define what they want to annotate. First of all time annotations have been incorporated. This is a technique used in all of the existing solutions that provide annotation services. Additionally we have also implemented the possibility to annotate a certain area on a slide. Now the time annotations receive an extra dimension as is can indicate more clearly what exactly is annotated.

Furthermore, the SynergE-Learning Web Portal supports annotating existing annotations. This way annotations can become whole discussion threads that all reference the same annotation object, where in existing tools only new annotations could be made. This technique improves the student interaction with lectures. Lecturers can pose question on certain topics and receive immediate feedback. And students can receive different angles on subjects from their peers.

Annotation Content

The annotation service of the SynergE-Learning Web Portal allows annotations to consist of rich text content. Other implementations only provide plain text as an annotation. This is because existing solutions do not see annotation services as a main part of their software. They assume that annotations are not made often and if they are made, that they should be small and concise.

In our opinion, annotations are a crucial part of enhancing the students educational experience. We attempt to make lectures more interactive, even after they have been given. When giving the students the opportunity to contribute to a lecture, they should have access to all the necessary means. As a first, we allow students to make rich content annotations. They can personally lay out their annotations, change the font, include lists and tables. Furthermore, we also allow users to add files to their annotations. This can be an supporting image or video file, similar other presentations or documents about related work.

Annotation Sharing

We previously mentioned that students can create annotations and share these annotations. However, as privacy has become a widely discussed issue, we have incorporated a sharing system where users can define with whom they wish to share annotations with. Existing tools are also aware of privacy

issues, but do not provide flexible sharing. Annotations are either publicly available or can only be accessed by the creator of the annotation.

Our implementation introduces user groups for sharing annotations. Students can create their own user groups and add other users to their likings. Once a group has been created, students can select this group to share their annotations with. Upon creating an annotations, the privacy settings need to be entered. Users can choose between `Public`, `Private` or `Share with Group`. Once the `Share with Group` option has been chosen, students can select the group that they wish to share the annotation with.

Visualising Annotations

Providing several annotation methods is one thing, but there is also a great need to visualise them properly. Other lecture capturing tools show annotations as a plain list of annotations accompanied with their timestamp. Only Tegrity also incorporates visualisation on the media controller.

The SynergE-Learning Web Portal has implemented the list view of the annotations as well. However, this list is able to directly visualise the rich content. Furthermore, we have extended our media controller with marks indicating when annotations are made. In the case that an on-slide annotation is triggered by the media controller, it is automatically visualised on the slide, indicating the selected area of the annotation. We provided a system that is flexible, students can change the playback setting to their likings, if they wish to disable the on-slide annotations, they can be disabled and will be treated as regular off-slide annotations.

Ingest and Process Methods

All of the existing solutions build on the same idea of capturing the screen of the presenter to record a lecture. This technique allows great flexibility in recording different input. However, this demands large file processing once we want to make it available for reviewing. First, the recording needs to be uploaded and a recording of an hour can take in several hundreds of megabytes. And after it is uploaded, the recording needs to be analysed as well to extract lecture information such as slide transitions. This requires lots of processing power and time.

The SynergE-Learning Web Portal provided a lightweight solution. A lot of the processing is done during the presentation recording. The PaperPoint technology logs all of the actions during a presentation. This makes processing a recorded lecture by the web portal much easier. The SynergE-Learning Web Portal can reconstruct a whole recording with just the presentation file and the PaperPoint logfile. When we take look at the file size of the logfile for a recording of an hour, we can see that it only takes a couple of megabytes. Processing the logfile on the server side does not ask for any complicated and time consuming algorithms.

	SynergE-Learning Web Portal	Tegrity	Panopto	Elluminate	Echo360	Sonic Foundry	Matterhorn
slide overview	Linear/ enhanced linear/path visualisation	linear	linear	✖	linear	linear	linear
transition marks	✓	✓	✓	✓	✖	✖	✓
annotations	Rich text, documents, audio/video	basic media	basic notes	✖	✖	✖	basic media
annotation sharing	Public/private/ group sharing	public/private	public/private	✖	✖	✖	unknown
interaction	Collaborative annotation	chat	question board	chat - whiteboard collaboration	✖	questions and polls	✖
ingest method	PaperPoint XML file analyses	screen capture	screen capture	screen capture	screen capture	screen capture	screen capture
process method	PaperPoint processing and on-server analyses	on-server processing	on-server processing	on-server processing	on-server processing	on-server processing	on-server processing
conclusion	Overcomes most shortcomings of others	best experience	replay-oriented	business and webcast-oriented	replay-oriented	business and webcast-oriented	work in progress

Figure 5.1: Comparing the SynergE-Learning Web Portal to exiting solutions

Chapter 6

Conclusion and Future Work

The goal of this thesis was to overcome the depersonalisation of the educational system that is currently present. We wanted to actively involve students in their own learning experience. We have introduced the SynergE-Learning Web Portal, where students can interact with their lectures and fellow students. Our implementation adds an extra dimension to e-Learning as students are now also able to provide content and enrich lectures. The vision of students on certain topics can be insightful for other students to obtain a better understanding of the the lecture material. It can also provide feedback to lecturers, giving them information about difficulties in the presented lectures. Our system provides solutions on shortcomings of existing lecture capturing tools:

- It provides a lightweight processing mechanism;
- The system provides clear lecture overviews for linear and non-linear presentations;
- A flexible annotation services has been incorporated;
  - Various annotation methods are possible;
  - Diverse annotation content can be added;
  - Adjustable annotation visualisation settings;
- Highly configurable privacy and sharing options.

Future Work

In this thesis, we have introduced a web portal for reviewing lectures. This implementation has been created based on the Flex framework, with the idea that there is little to no work to port the web portal from the browser to the desktop and to mobile devices. The Flex framework is capable of doing the conversion from a common code base. Adjusting the code for a desktop application should be straightforward. To port the web portal to mobile applications requires a new UI to be created. However, most of the code can be reused. Existing solutions promote their functionality that recorded lectures are also converted to mobile device friendly file formats during the upload and processing phase. We can take this a step further in the sense that not only lectures can be reviewed, but also annotations and overviews can be accessed via mobile applications.

In the future, the SynergE-Learning Web Portal could also incorporate live-broadcasting and streaming options. Lots of the existing solutions already provide this functionality, mostly because the lecture capturing systems are built around live-broadcasting technologies. In literature, Flex is often linked to the Adobe Flash Media Server to provide streaming functionalities. However, we would like to keep this solution completely open source. Red5 Media Server[1] is a powerful open source video streaming solution to the Adobe Flash Player. It is based on Java and other powerful open source frameworks. Red5 Media Server can be dispatched on a Tomcat server. As our implementation also runs partially on a Tomcat, the Red5 Media Server would be the perfect solution to integrate in the SynergE-Learning Web Portal. Once the Red5 Media Server is incorporated in the architecture, the web portal should be extended with some additional components handling the lecture streaming.

[1] `http://www.red5.org/`

Appendix A

Installation

In this appendix, we will lay out the different steps that are needed to install the SynergE-Learning Web Portal. The web portal requires a web server that supports PHP, a MySQL database and a Tomcat server for the JODConverter web service.

A.1 Database Setup

First we will set up the MySQL database. Listing A.1 shows the SQL code that needs to be executed to create the tables in the database. Furthermore, an administrator account will be created for the first use. The login and password for the administrator account are respectively `admin` and `root`.

Listing A.1: SQL Code for the MySQL Database

```
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

-- ------------------------------------------------------------------


--
-- Tablestructure for table `portal_annotation`
--

CREATE TABLE IF NOT EXISTS `portal_annotation` (
  `annotation_id` int(11) NOT NULL AUTO_INCREMENT,
  `event_id` int(11) NOT NULL,
  `parent_id` int(11) DEFAULT NULL,
  `content` text NOT NULL,
  `XCoord` int(11) DEFAULT NULL,
  `YCoord` int(11) DEFAULT NULL,
  `annot_author` int(11) NOT NULL,
  PRIMARY KEY (`annotation_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

```sql
--
-- Tablestructure for table `portal_courselist`
--

CREATE TABLE IF NOT EXISTS `portal_courselist` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `course_id` int(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

-- ------------------------------------------------------------------------

--
-- Tablestructure for table `portal_courses`
--

CREATE TABLE IF NOT EXISTS `portal_courses` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `description` text,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

-- ------------------------------------------------------------------------

--
-- Tablestructure for table `portal_events`
--

CREATE TABLE IF NOT EXISTS `portal_events` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `lecture_id` int(11) NOT NULL,
  `type_id` int(11) NOT NULL,
  `startTime` int(11) NOT NULL,
  `endTime` int(11) DEFAULT NULL,
  `slideNumber` int(11) DEFAULT NULL,
  `author_id` int(11) NOT NULL DEFAULT '0',
  `share_type` int(11) NOT NULL,
  `share_id` int(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

-- ------------------------------------------------------------------------

--
-- Tablestructure for table `portal_eventtypes`
--
```

```
CREATE TABLE IF NOT EXISTS `portal_eventtypes` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `type` varchar(30) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

-- ------------------------------------------------------------------

--
-- Tablestructure for table `portal_groups`
--

CREATE TABLE IF NOT EXISTS `portal_groups` (
  `group_id` int(11) NOT NULL AUTO_INCREMENT,
  `groupName` varchar(30) NOT NULL,
  `groupOwner` int(11) NOT NULL,
  PRIMARY KEY (`group_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

-- ------------------------------------------------------------------

--
-- Tablestructure for table `portal_lectures`
--

CREATE TABLE IF NOT EXISTS `portal_lectures` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `course_id` int(11) DEFAULT NULL,
  `author_id` int(11) DEFAULT NULL,
  `name` varchar(50) NOT NULL,
  `date` date DEFAULT NULL,
  `length` int(11) DEFAULT NULL,
  `presName` varchar(50) DEFAULT NULL,
  `videoName` varchar(50) DEFAULT NULL,
  `logName` varchar(50) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

-- ------------------------------------------------------------------

--
-- Tablestructure for table `portal_penannotations`
--

CREATE TABLE IF NOT EXISTS `portal_penannotations` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `event_id` int(11) NOT NULL,
  `pointColl` text NOT NULL,
  `ocr` text,
  PRIMARY KEY (`id`)
```

```
) ENGINE=InnoDB  DEFAULT CHARSET=latin1  AUTO_INCREMENT=1 ;

-- ------------------------------------------------------------------

--
-- Tablestructure for table `portal_roles`
--

CREATE TABLE IF NOT EXISTS `portal_roles` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `type` varchar(20) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1  AUTO_INCREMENT=4 ;

--
-- Data for table `portal_roles`
--

INSERT INTO `portal_roles` (`id`, `type`) VALUES
(1, 'student'),
(2, 'admin');
(3, 'prof');

-- ------------------------------------------------------------------

--
-- Tablestructure for table `portal_usergroups`
--

CREATE TABLE IF NOT EXISTS `portal_usergroups` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `group_id` int(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1  AUTO_INCREMENT=1 ;


-- ------------------------------------------------------------------

--
-- Tablestructure for table `portal_users`
--

CREATE TABLE IF NOT EXISTS `portal_users` (
  `user_id` int(11) NOT NULL AUTO_INCREMENT,
  `role_id` int(11) NOT NULL,
  `firstName` varchar(30) DEFAULT NULL,
  `lastName` varchar(30) DEFAULT NULL,
  `login` varchar(30) NOT NULL,
  `password` varchar(50) NOT NULL,
```

```
  PRIMARY KEY (`user_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1  AUTO_INCREMENT=2 ;

--
-- Data for table `portal_users`
--

INSERT INTO `portal_users`
(`user_id`, `role_id`, `firstName`, `lastName`, `login`, `password`) VALUES
(1, 2, 'Admin', 'Admin', 'admin', MD5('root'));
```

A.2 PHP, PEAR and IMagick Extension

For the server-side application logic, the SynergE-Learning Web Portal depends on PHP. The currentl implementation runs on PHP Version 5.3.5. We also make use of the IMagick Extension for converting the PDF file of the presentations to images as explained in Section 4.2.2.

IMagick is a native PHP extension to create and modify images using the ImageMagick API. So first, we need to install ImageMagick. The files for installing ImageMagick, as well as documentation can be found on the ImageMagick website[1]. Once ImageMagick is installed, we can download the extension for PHP and install it. The files can be found on PECL[2], which is a repository for PHP Extensions.

To install PHP extension, we need to install PEAR, PHP Extension and Application Repository, on our PHP server. Once PEAR is implemented, we can go ahead and extend PHP with IMagick. We will also need to install some dependencies for JODConverter later on. PEAR can be found on its PHP page[3], together with some installation guidelines. Basically we need to download the `go-pear.phar` file and place it on the server and run it. Once this is done, we need to edit the `php.ini` configuration file to add the path to the PEAR installer. Listing A.2 shows the required code.

Listing A.2: PHP.ini code

```
include_path=".;C:\wamp\bin\php\php5.3.5\PEAR";
```

Now we can install the IMagick extension via PEAR by placing the previously downloaded IMagick files in the PEAR folder and running `pear install imagick` from the command line. We will need to do the same for another extension for JODConvertor. First we will need to download the `HTTP_Request` files and, since this extension depends on the `Net_URL`, we will also need to download these files. Once they are placed in the PEAR folder, they can be installed via the command line. The files and detailed instructions can be found on the PEAR website[4].

A.3 JODConverter

JODConverter is used to convert presentations to PDF, which IMagick will convert to images. JODConverter provides a web service that runs on a Tomcat Server. It will call the OpenOffice.org service to convert files. First, we will need to install the OpenOffice.org service on the server. OpenOffice.org

[1] `http://www.imagemagick.org/script/binary-releases.php`
[2] `http://pecl.php.net/package/imagick`
[3] `http://pear.php.net/package/PEAR`
[4] `http://pear.php.net/package/HTTP_Request` and `http://pear.php.net/package/Net_URL/`

can simply be downloaded via their website[5] and be installed as any other program. Once the installation is complete, we can continue to set up the Tomcat server. On the website of JODConverter[6] we can download the necessary files. We can either choose to download a pre-configured Tomcat server, or a WAR file to deploy on an existing Tomcat server. The only thing that remains now is to adjust the reference to the Tomcat server in the `upload.php` file[7], which can is responsible for querying the server to convert the files.

A.4 SynergE-Learning Web Portal

Now that every dependency is in place and configures, we can continue with deploying the SynergE-Learning Web Portal. All we need to do is place the content of the `LecturePortal` on our web server. Now we can create a simple HTML web page to embed the `SWF` file. When we navigate to this HTML page, the web portal will be loaded and we can make use of all its functions.

[5] `http://www.openoffice.org/`
[6] `http://sourceforge.net/projects/jodconverter/`
[7] This file can be found in the source files under the php folder

Bibliography

[1] Anoto AB. `http://www.anoto.com`.

[2] Robert A. Bartscha and Kristi M. Cobernb. Effectiveness of PowerPoint Presentations in Lectures. *Computers and Education*, 41(1):77–86, June 2003.

[3] Alan D. Greenberg and Andy Nilssen. Personalization: The Next Evolution in Lecture Capture Technology. White Paper, August 2010.

[4] Thomas Heß. Fly, Expressive and Conveying Planar Presentations. Master's thesis, University of Trento and RWTH Aachen University, March 2011.

[5] Educause Learning Initiative. 7 Things You Should Know About Lecture Capture. White Paper, December 2008.

[6] Educause Learning Initiative. 7 Things You Should Know About Collaborative Annotation. White Paper, October 2009.

[7] Educause Learning Initiative. 7 Things You Should Know About Next Generation Presentation Tools. White Paper, January 2010.

[8] Educause Learning Initiative. 7 Things You Should Know About Privacy In Web 2.0 Learning Environments. White Paper, September 2010.

[9] Markus Ketterl, Olaf A. Schulte, and Adam Hochman. Opencast Matterhorn: A Community-Driven Open Source Solution for Creation, Management and Distribution of Audio and Video in Academia. In *Proceedings of ISM 2009, 11th IEEE International Symposium on Multimedia*, pages 687–692, San Diego, USA, December 2009.

[10] Joanne Patterson. Using Lecture Capture Technologies for Distance Learning: A Case Study Using Panopto. In *Proceedings of ASCILITE 2009. Same Places, Different Spaces*, pages 746–748, Auckland, New Zealand, December 2009.

[11] Brian E. Perron and Alyson G. Stearns. A Review of a Presentation Technology: Prezi. *Research on Social Work Practice*, 21(3):376–377, May 2011.

[12] Elke I. Reuss, Beat Signer, and Moira C. Norrie. PowerPoint Multimedia Presentations in Computer Science Education: What Do Users Need? In *Proceedings of USAB 2008, 4th Symposium on Usability & HCI for Education and Work*, pages 281–298, Graz, Austria, November 2008.

[13] Yong Rui, Anoop Gupta, Jonathan Grudin, and Liwei He. Automating Lecture Capture and Broadcast: Technology and Videography. *Multimedia Systems*, 10(1):3–15, 2004.

[14] Beat Signer. *Fundamental Concepts for Interactive Paper and Cross-Media Information Spaces.* PhD thesis, ETH Zurich, August 2005. ETH No. 16218.

[15] Beat Signer and Moira C. Norrie. As We May Link: A General Metamodel for Hypermedia Systems. In *Proceedings of ER 2007, 26th International Conference on Conceptual Modeling*, pages 359–374, Auckland, New Zealand, November 2007.

[16] Beat Signer and Moira C. Norrie. PaperPoint: A Paper-Based Presentation and Interactive Paper Prototyping Tool. In *Proceedings of TEI 2007, 1st International Conference on Tangible and Embedded Interaction*, pages 57–64, Baton Rouge, USA, February 2007.

[17] Beat Signer and Moira C. Norrie. A Model and Architecture for Open Cross-Media Annotation and Link Services. *Information Systems*, 36(3):538–550, May 2011.

[18] Jürgen Steimle, Oliver Brdiczka, and Max Mühlhäuser. Collaborative Paper-Based Annotation of Lecture Slides. *Educational Technology & Society*, 12(4):125–137, 2009.

[19] Edward R. Tufte. *The Cognitive Style of PowerPoint: Pitching Out Corrupts Within.* Graphics Press, Cheshire, USA, 2nd edition, January 2006.

[20] Nadir Weibel. *A Publishing Infrastructure for Interactive Paper Documents.* PhD thesis, ETH Zurich, July 2009. ETH No. 18514.

[21] Michelle Wilkerson, William G. Griswold, and Beth Simon. Ubiquitous Presenter: Increasing Student Access and Control in a Digital Lecturing Environment. In *Proceedings of SIGCSE 2005, 36th Technical Symposium on Computer Science Education*, pages 116–120, St. Louis, USA, February 2005.