Experiences from Developing Educational Presentation Systems

Georg Turban
Technische Universität Darmstadt
Germany
turban@informatik.tu-darmstadt.de

Max Mühlhäuser
Technische Universität Darmstadt
Germany
max@informatik.tu-darmstadt.de

Abstract

This paper is a report about experiences and observations during the development of educational presentation systems for higher education. The paper discusses workflows during a typical slide-centric presentation. It presents different models that can be hierarchically aligned and extended by temporal information in order to reflect the workflows and to structure the content presented during a lecture. While structure and timing contribute to the high-level design of presentation systems the paper also identifies aspects that are relevant for mid- to low-level design and implementation of components within presentation systems. With respect to the comparison of related work, four key aspects are discussed. Finally, we present conclusions for the general design and development of flexible presentations systems.

1. Introduction

Today, many different presentation systems are used for education. The systems comprise very simple up to more complex functionality and vary in their intention and technical realization.

Some systems such as Windows Journal [1], e-chalk [2] or TinyWB [3] focus on the development of content from scratch during a lecture, while other systems such as PowerPoint [1], Lecturnity [4] or Lectern II [5] focus on the presentation of prepared lecture slides. Systems such as Camtasia [6], TeleTeachingTool [7] or Presenter [8] process desktop captures that can be augmented. In addition, systems of the latter two groups often introduce features to support creating and annotating blank slides during a session, in order to overcome the disadvantages and limitations of slide-centric talks. This paper focuses on selected aspects of such systems and on our experiences and observations during the development of educational presentation systems.

In a top-down manner, we analyze the typical workflow in a slide-centric lecture and present hierarchically aligned models that are capable of representing the structure and temporal relationships within a lecture in chapter 2. Chapter 3 focuses related work in respect to highlighted aspects and their specific implementation. Based on the observations and our own experiences during the development of educational presentation systems, design issues for the discussed aspects are also proposed in chapter 3. The paper closes with chapter 4, a summary of the presented work.

2. Representing a Typical Lecture Scenario

This chapter focuses on presentation-centric lectures and discusses a typical lecture scenario in order to derive a suitable model-based representation for a presentation system that is capable to reflect workflows in respect to processed content.

2.1. A Typical Lecture Scenario

In presentation-centric lectures, the lecturer mediates content using presentations. During a lecture series, several presentation files are created for this purpose and presented to the audience. A presentation consists of different slides that are often augmented using digital ink. Afterwards, the modified presentations are usually distributed and reused by the students.

2.2. Workflows during a Lecture

The sequence diagram in figure 1 shows the timelines of different types of models (dashed lines) and the period of their modification during a typical lecture that relies on slide-based presentations.

A so-called *lecture session* is initialized and terminated by using a presentation system. During a session, slides of different presentations are presented

0-7695-3084-2/07 $25.00 © 2007 IEEE
DOI 10.1109/ISM.Workshops.2007.67

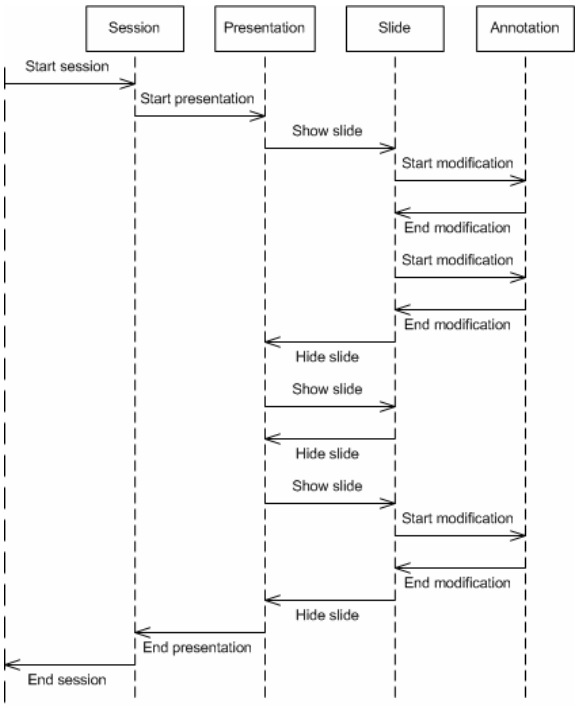

Figure 1: Sequence diagram that depicts the timeline of modifications, applied to different types of models, during a lecture.

and augmented. The beginning and ending of modifications result from different actions such as *show slide* and *hide slide*. Actions occur on different stages during a session and can be distinguished by their granularity.

2.3. Timed Models for the Representation

The different stages within a workflow can be represented using a hierarchical set of models that consists of models for sessions, presentations, slides and annotations. The hierarchical alignment and temporal relationships can be expressed using the alignment of models shown in figure 2. Figure 2 also contains *timing*-models that are capable to reflect the temporal information of processes within the workflows. The hierarchy consists of following models.

Session: A *session-model* consists of meta-information and references to *presentation-models*. Examples for meta-information are *the name of the lecturer, the name of the lecture series* and *the date, time* and *room of the lecture*.

Presentation: *Presentation-models* consist of meta-information such as *the title of a presentation* and references to *slide-models*.

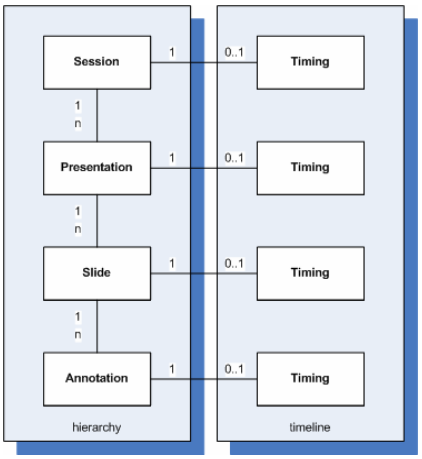

Figure 2: Hierarchically aligned models.

Slide: A *slide-model* consists of a *title* and *content* that is in many cases just a simple image, but may also contain *speaker notes*. The model can contain further references that, e.g., point to *annotation-models*.

Annotation: An *annotation-model* represents different types of annotation. In this work, we focus on freehand ink annotations.

Timing: A *timing-model* is capable of representing the beginning and ending of a models modification. The corresponding time span does not essentially match the lifecycle of the associated object. For instance, slide-models may be created (but still not modified) in order to provide previews for navigation purposes.

3. Selected Aspects of Presentation Systems

Different presentation systems are focused in respect to their intention and underlying technical realization in the following sections. Based on the discussion, the application domain is decomposed into selected aspects and components that can be (re-)used in order to ease the development of similar systems and to increase the understanding for the development of such systems.

3.1. Static vs. Dynamic Content

The requirements and complexity of systems and their implementations varies heavily based on whether the presentation of static or dynamic content is supported. Presenting and augmenting static content can be reduced to the problem of processing images. Many systems including Classroom Presenter [9] and Multimedia Lecture Board [10] chose this approach and therefore require converting presentation files into

image sets. Annotating dynamic content is comparable to annotation of videos [11], but requires additional, expensive processes like screen-capturing and real-time rendering to various destinations, such as the displays of lecturers and students.

A static slide can be represented by a single image. In contrast, animated slides or videos from systems that follow the screen-recording approach produce far more images. For most applications, dealing with over 25 images per second is unfeasible. Especially distributed solutions process (rectangular) updates of screen regions and limit the overhead using the remote frame buffer protocol. A suitable solution is to add the updates including timestamps to the slide-model or to introduce a new type of model as shown in figure 3 and named *update-model*.

For collaborative video annotations (rather text than ink), Vannotea [12] proposes the following meta-model: video nodes consist of video segments. A segment contains a single key frame and temporal information. While video and video segment nodes can be compared with our presentation and slide models, a list of key frames (instead of a single key frame) are comparable to our slide updates.

Update: An *update-model* is associated with a slide-model and represents (regional) updates. It also contains a reference to a *timing-model* that reflects the time span or moment when the update occurred. Those timings are especially valuable for later playback. Similar to common video compression methods like MPEG, the update region can also be an update of the whole slide.

Partial updates also increase the performance during rendering and transmission of the content, while tasks such as navigation and previewing require a "full image". If only partial updates are processed, recalculating the image representation of a specific, queried moment becomes expensive. Therefore, systems such as [7], [8] and [13] occasionally force full updates of the whole framebuffer, similar to I-frames in

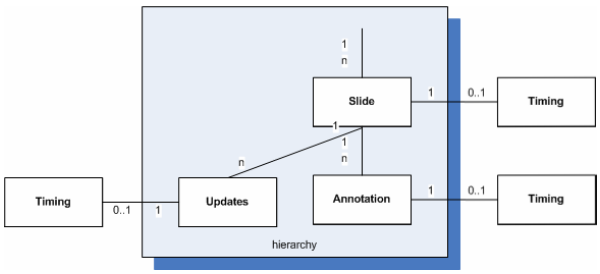

Figure 3: Enhancing the slide-model with a model capable to represent (regional) slide updates.

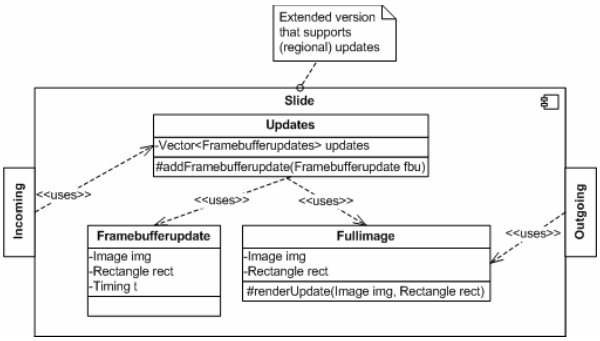

Figure 4: Extended processing unit for (regional) slide updates.

MPEG encodings. In addition, we experienced significant ease of handling updates by introduction of the component shown in figure 4. Please note that we do not discuss in detail that detected modifications of the source frame buffer can be optimized by the specific implementation (e.g., VNC) and split into a sequence of transmitted, regional updates. From a black-box view, the component consumes partial updates, but delivers a full image to other components. The full image is always up to date and can be spontaneously used for storage and previewing purpose. From a white-box view, the component receives and stores (partial) frame buffer updates that are continuously rendered on top of an image that can be queried from other components via an outgoing port.

3.2. Local vs. Distributed Systems

Systems that support dynamic content vary in the way how the content is obtained. There are large differences between the complexity and performance of such solutions.

The distributed presentation scenario in figure 5 corresponds to the work of the University of Cambridge [13] and TeleTeachingTool. The lecturer prepares his content before the lecture on his own system. During the lecture, a remote connection to the

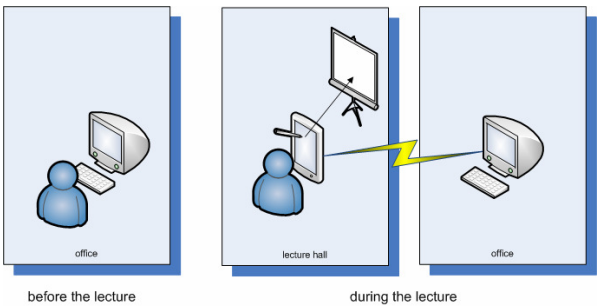

Figure 5: Distributed presentation scenario.

system located in his office is established using a protocol such as VNC. The remote desktop is then continuously mirrored to the local presentation system that offers annotation functionality. The remote system can be controlled using the keyboard and the mouse.

In contrast to the approach of the presented two systems, Presenter uses virtual frame buffers that have been developed in a former project state and can be installed on the system of the lecturer. The desktop is extended and content located on the virtual frame buffers can be processed like depicted in the former example. In contrast to the distributed approach, a second system is not required and there are fewer pitfalls in developing a local screen mirror.

By extending the slide-model we introduced support for incremental updates of images that are able to represent the frequently updated images that are usually delivered by the presented approaches. The general processing of updates, whether obtained from local or remote desktops, is identical.

The event handling between local and remote systems is bidirectional and varies in contrast to the image handling. Controlling remote desktops requires mapping local mouse and keyboard actions, e.g., provided by the frame buffer protocol. The mapping of keyboard actions can be omitted on windows systems that contain virtual desktops while mouse actions must be still handled. Regarding the information flow between the systems, (local) input events on the remote system directly force frame buffer updates that are transmitted to the local system, e.g. the remote frame buffer protocol transmits mouse movements separately. But due to the loose coupling of local and remote systems, presentation-centric actions such as slide-transitions are usually not handled and transmitted.

There are different approaches that e.g., use software installed on the local system to generate and inject messages by extending the common protocols. Other approaches avoid installing software on the local system and must therefore post-process images so that for instance, slide transitions can be detected. There are even approaches that avoid any modification of the remote system, but are also able to provide benefits, while the lecturer can still operate his system locally. Systems such as TeraVision [14], T-Cube [15] and ProjectorBox [16] have been primarily developed for recording purposes or transmission of video signals. But approaches that are only coupled using the video signal of computers are very limited: it is rather difficult to support ink annotations, for instance. Annotations on the local system would require clearing them manually after slide transitions and to operate the local system. Solutions have been developed that even redirect mouse and keyboard events using hardware. We expect that such solutions will still remain very limited in their functionality, but may get more popular for scenarios that focus on seamless recording support.

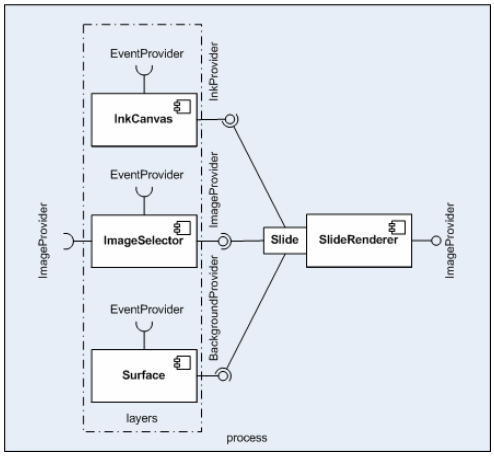

Figure 6: Component-based, layered rendering of annotations on top of static or dynamic content.

3.3. Annotations

In contrast to the differences regarding the processing of static or dynamic content, augmenting the content is usually implemented very similarly. Annotations like ink annotations are added on top of the content and remain visible until the user or dedicated events such as slide-transitions force them to disappear.

Annotations are typically not adapted (e.g., they are not automatically translated or scaled) in correspondence to the underlying content. Associating the underlying content is rather difficult and contains problems such as object recognition. A restricted solution is presented by *Avaya* [17] which associates annotations on top of HTML-websites using the underlying DOM-tree. But current presentation systems, especially the ones supporting dynamic content, do not focus on associations. Therefore, annotations can be rendered independently of the type of the underlying content and are implemented efficiently using layers that are well-known from 2d graphic applications. Programming languages typically provide overlays that are similar to glass panes and their processing is often hardware-accelerated.

Our subsystem for core presentation support consists of the components presented in figure 6. By decoupling the annotation layer from the underlying content, we can dynamically switch between different image sources and support annotations on surfaces similar to black- or whiteboards, static images and dynamically updated images sources. The subsystem

can be configured to support different scenarios based on different image-sources. A lightweight whiteboard-setup is for instance used by TinyWB [3]. Our image layer also enables the corresponding functionality of systems that follow the approach of annotating static images and those that follow the approach of annotating video streams. In section 3.2., we presented examples for systems that follow different approaches. The layering turned out to be very flexible and is not limited to the presented scenarios. Additional layers can be added on top of the lecturer's annotation layer and support private annotations or collaborative scenarios while handling annotations of the audience. Private annotations can also be created before the lecture; in [18] they are called *instructor notes* and discussed in more detail.

3.4. Events vs. Content

The assignment of annotations to slides and removal of annotations during slide-transitions is a major difference for systems that support static vs. dynamic content. Systems that convert content and work on image sets can obviously easily handle the content and generate or process the required events. Systems that process arbitrary dynamic content initially can not rely on any events available within the underlying presentation system.

Systems such as TeleTeachingTool use key-bindings that are usually associated with the underlying actions in presentation systems such as page up/down for seeking to the next/previous slide. Systems like virtPresenter [19] or Presenter use add-ins for PowerPoint to receive the required events. Presenter contains also a different, more *proactive* approach: an extension for presenter is capable to load and control PowerPoint-presentations and can therefore handle events easier and avoids modifying PowerPoint. A uniform way to handle any slide-based presentation, the corresponding reference implementation called Universal Presentation Controller and the specific handling of PowerPoint are discussed in [20].

We developed a couple of different solutions to control PowerPoint or to subscribe to PowerPoint events. The different implementation had a significant impact on our event handling. For instance, latter solution requires handling the delivered PowerPoint events asynchronously. Our other solutions – PowerPoint-Controller and Universal Presentation Controller – provide different controls on the user-interface for navigation support. Different action listeners are associated with those GUI-controls and events can be handled by our presentation system in

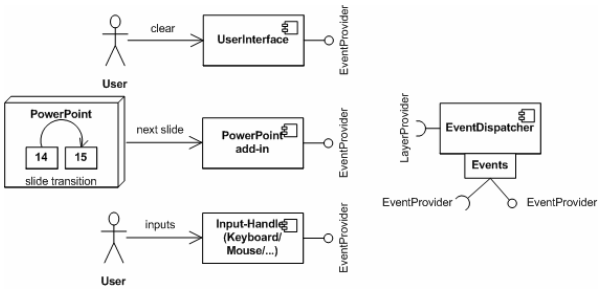

Figure 7: An event dispatcher unifies handling of control flows and decouples components.

advance to the requested slide transitions. After the presentation system returns from the specific processes (e.g., creating of snapshots and new slide-models) the required operations are forwarded to PowerPoint.

We identified different events such as *clear*, *store* and *next slide* and modified the event handling by introducing a dedicated event-dispatcher like shown in figure 7. One major advantage is that the core presentation subsystem depicted in figure 6 can now be developed independent from modifications to event-providers like the ones shown on the left side of figure 7. This also applies to handling of user inputs using the mouse and keyboard. Such inputs can be delivered to the event-dispatcher that either consumes or forwards them to a specific layer.

4. Summary

We presented our experiences and observations during the development of educational presentation systems for higher education. The paper discussed workflows during a typical slide-centric presentation and presented hierarchically aligned and timed models. While structure and timing contribute to the high-level design of presentation systems, the presented aspects are relevant for mid- to low-level design and implementation of components. The key aspects were discussed in respect to related work. Finally, we presented solutions and experiences for the identified aspects that facilitated the development of flexible educational presentations systems.

References

[1] Microsoft Corporation, "Microsoft Product Information Center," http://www.microsoft.com, last visited July 31st, 2007.

[2] L. Knipping, "An Electronic Chalkboard for Classroom and Distance Teaching. Ph.D. thesis, Fachbereich Mathematik und Informatik", Freie Universität Berlin, Germany, 2005.

[3] G. Turban and M. Mühlhäuser, “An Open Architecture for Face-to-Face Learning and Its Benefits”, in: Proceedings of the 8th IEEE International Symposium on Multimedia, San Diego, CA, USA, 2006.

[4] W. Hürst, R. Müller and T. Ottmann, “The AOF Method for Production, Use, and Management of Instructional Media”, in: Proceedings of the International Conference on Computer in Education, Melbourne, Australia, 2004.

[5] N. Joukov and T. Chiueh, “Lectern II: A multimedia lecture capturing and editing system”, in: Proceedings of the International Conference on Multimedia and Expo, Baltimore, Maryland, USA, 2003.

[6] TechSmith Corporation, “Camtasia Studio Screen Recording and Presentation”, http:// www.techsmith.com, last visited, July 31, 2007.

[7] P. Ziewer and H. Seidl, “Transparent TeleTeaching”, in: Proceedings of the Australasian Society for Computers in Learning in Tertiary Education conference, Auckland, New Zealand, 2002.

[8] G. Turban and M. Mühlhäuser, “A Framework for Educational Presentation Systems and its Application”, in: Proceedings of the 1st ACM Workshop on Educational Multimedia and Multimedia Education in conjunction with ACM Multimedia 2007, Augsburg, Bavaria, Germany.

[9] R. Anderson, R. Anderson, L. McDowell and B. Simon, “Use of Classroom Presenter in Engineering Courses”, in: Proceedings of the 35th Annual Conference on Frontiers in Education, 2005.

[10] J. Vogel, “Präsentation und Kollaboration in Televeranstaltungen mit dem multimedia lecture board”, in: Tagungsband der 17. DFN-Arbeitstagung über Kommunikationsnetze, LNI, GI, Düsseldorf, Germany, 2003.

[11] D.C.A. Bulterman, “Creating Peer-Level Video Annotations for Web-Based Multimedia”, in: Proceedings of Eurographics Workshop on Multimedia, 2004.

[12] R. Schroeter, J. Hunter and D. Kosovic, “Vannotea - A Collaborative Video Indexing, Annotation and Discussion System For Broadband Networks”, in: Proceedings of Knowledge Markup and Semantic Annotation Workshop, K-CAP 2003, Sanibel, Florida, USA, 2003.

[13] S.F. Li, M. Spiteri, J. Bates and A. Hopper, “Capturing and Indexing Computer-based Activities with Virtual Network Computing”, in: Proceedings of the 2000 ACM Symposium on Applied Computing, Como, Italy, 2000.

[14] J. Leigh, J. Girado, R. Singh, A. Johnson, K. Park and T.A. DeFanti, “TeraVision: a Platform and Software Independent Solution for Real Time Display Distribution in Advanced Collaborative Environments”, Electronic Visualization Laboratory, University of Illinois at Chicago, 2002.

[15] M. Ma, V. Schillings, T. Chen and C. Meinel, “T-Cube: A Multimedia Authoring System for eLearning”, in: Proceedings of E-Learn 2003, Phoenix, Arizona, USA, 2003.

[16] L. Denoue, D. Hilbert, J. Adcock, D. Billsus and M. Cooper, “ProjectorBox: Seamless presentation capture for classrooms”, in: Proceedings of E-Learn 2005, Vancouver, Canada, 2005.

[17] R. Kashi and S. Ramachandran, “An Architecture for Ink Annotations on Web Documents”, in: Proceedings of the 7th International Conference on Document Analysis and Recognition – Vol. 1, Edinburgh, Scotland, 2003.

[18] B. Simon, R. Anderson and S. Wolfman, “Activating Computer Architecture with Classroom Presenter”, in: Proceedings of the Workshop on Computer Architecture Education, in conjunction with the 30th International Symposium on Computer Architecture, San Diego, CA, USA, 2003.

[19] R. Mertens, H. Schneider, O. Müller and O. Vornberger, “Hypermedia Navigation Concepts for Lecture Recordings”, in: Proceedings of E-Learn 2004. Washington, DC, USA, 2004.

[20] G. Turban and M. Mühlhäuser, “A Uniform Way to Handle Any Slide-Based Presentation: The Universal Presentation Controller”, in: Advances in Multimedia Modeling. 13th International ACM Multimedia Modeling Conference, Singapore, 2007.