An open architecture for face-to-face learning and its benefits

Georg Turban
Darmstadt University of Technology
Germany
turban@informatik.tu-darmstadt.de

Max Mühlhäuser
Darmstadt University of Technology
Germany
max@informatik.tu-darmstadt.de

Abstract

We already had the opportunity to present our category-based concept for the support of face-to-face learning during the last symposium. In the meantime, we made further experiences that contributed to the refinement of our concept and given publication.

Our last paper served to present the technical basis of our concept for the development of an open architecture and is recommended to be read, since we now present the results and benefits of its application in the classroom [1]. We will examine not only the benefits for lecturers and learners, but also provide examples that illustrate our contributions for developers and supporters.

The discussion of our results will be assisted by the presentation of features, use-cases and selected results of our continuously performed evaluation. All of them will be used to demonstrate the advantage and the power of our contribution that enabled us to adopt valuable features of other systems, to integrate completely new features and to combine our systems even with external ones.

1. Introduction

We developed an open architecture according to our goals to meet the requirements of many potential users and environments in the context of face-to-face learning. We are currently able to serve several national and international universities based on this open solution.

The underlying technical concepts and their realization have been motivated by several requirements analyses and mainly the following two development strategies:

In a top-down process, we used the results of our requirements analyses to derive essential functionality we decided to realize within the core of the system. To support flexibility and interoperability, we used the bottom-up strategy and developed several small units specialized in processing multimedia content in the focused context. We named them *media-modules* and designed them to be loosely coupled and highly reusable.

For us, the development of an open system is a continuous process and an open system should accomplish the following criteria: the assisted possibility…

- to easily extend its functionality
- to integrate external components or systems
- to be integrated into external systems or heterogeneous environments

In our opinion it is possible to order these criteria by the difficulty to support them within the whole process of creating an open architecture (see figure 1).

Our architecture relies on well defined models for several kinds of multimedia data like presentations, slides, annotations, annotation tools, display properties, etc. and several (context-based) views. In contrast to common approaches, we decided to introduce *working in a context* which for example assists the lecturer while operating different tasks. Instead of handling several individual applications he can be provided with so-called context views that are currently represented as tabbed panes. Besides advantages like uniform appearance and handling, contexts and their underlying models are able to cross-communicate and notify the users via ambient displays. The remaining functionality contains controllers and forms our *core*.

Regarding the vertical architecture presented in figure 2, we created a plugin-framework that resides in parallel to the introduced components at different levels of the architecture. The plugin-framework spans several layers because it for example contributes very

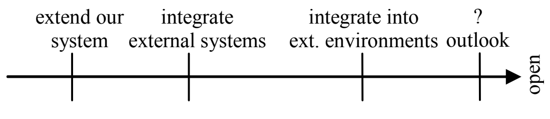


Figure 1: Moving toward an open architecture

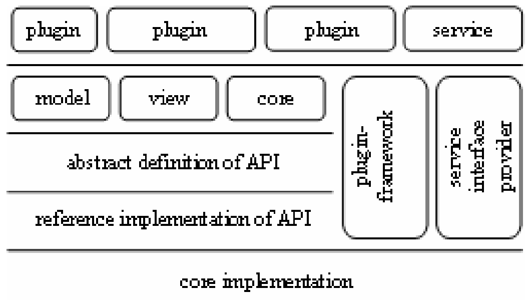

plugin
plugin
plugin
service
model
view
core
abstract definition of API
reference implementation of API
plugin-framework
service interface provider
core implementation

Figure 2: Vertical view of our open architecture

specific functions that provide developers with a plugin-hub for testing and developing purpose. In addition, we created a comprehensive documented API and identified extension points for pluggable components. Our support of plug-ins is *contract-driven*. Usually a specific contract is mainly defined by a set of interfaces that need to be implemented. This approach supports developers who use an IDE that provides them automatically with the corresponding stubs for methods and is able to detect basic problems already at compile time.

The identification of appropriate extension points and development of corresponding slots and sample plugins was focus by us and agrees to the design-principles for APIs and SPIs presented in [2] and [3].

Since our open architecture is designed and implemented to be platform independent (here, in Java 5), but developers who use a platform independent language occasionally get into situations where they can't avoid the integration of native code, we have been searching for a mechanism that allows us to integrate platform specific realizations.

According to our environments, these are especially issues related to performance that led us to replace platform independent solutions through native ones that e.g. support the windows platform. Although it may be possible to use our plugin-mechanism for such integrations, we favor an approach at this point that considers more the intention and characteristics for such replacements and fits better into our concept.

An approach that complies with our vision is the *integration of services that rely on a service provider interface*. A comparison of different scenarios that make use of this approach can be found in [3]. SPIs and services allow us a conceptually deeper integration and the modification of the core. Furthermore, the special characteristics like unparallel existence and permanent availability (especially outside the runtime of plugins) can be supported and expressed more strongly.

2. Results

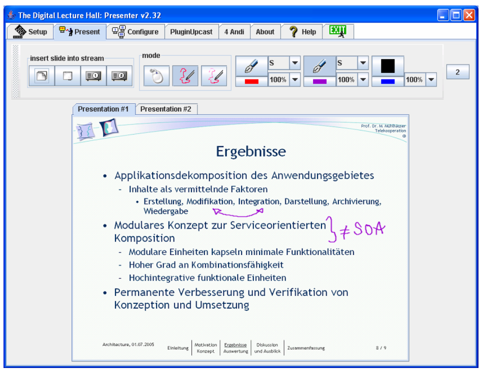
Our scenario of face-to-face learning is multifarious and covers typical areas like presentation, interaction, recording and playback of lectures. We present examples from named areas and highlight the benefits of our work for developers, users, participants and (either technical or assisting) supporters. The examples have been selected to cover all criteria and preferably to illustrate benefits for all named parties.

2.1. Extend presentation facilities

Classroom presenter contains an interesting feature that allows the lecturer to (temporarily) gain additional space for annotations on demand, which is based on the idea to shrink the currently presented content and is discussed in [4] and [5]. Because this idea overlaps with our concept of *history slides,* we integrated a similar feature called *add-notes* into our presentation

Sample 1: Simplified implementation of *add-notes*

```
package framework.plugin.image.process;

[…]

// Contracts
import framework.contract.plugin.meta.Base;
import framework.contract.plugin.view.Context;
import […].contract.plugin.slide.SlideControl;

// Controllers
import […].controller.slide.SlideController;

// Core-features
import framework.core.ImageViewer;

public class AddNotes
  implements Base, Context, SlideControl
{
  // Base (basically meta-infos like name of
  // author and name/description of component
  […]

  // Context (basically buttons to invoke the
  // functions to shrink and expand the slide)
  […]

  // Slide-control
  private SlideController  sc;

  public void setController(
    SlideController sc) { this.sc = sc; }

  // Core functionality that relies on
  // composition of existing media-modules
  private Slide originalSlide;

  public void shrinkSlide() {
    originalSlide = sc.getSlide();
    Image i0 = sc.historize(); // store&render
    Image i1 = Transformer.tf(i0,0.75f,0,0);
    ImageViewer.show(i1,Color.WHITE);
  }

  public void expandSlide() {
    sc.historize();  // store&render
    ImageViewer.hide();
    sc.setSlide(originalSlide);
  }
}
```

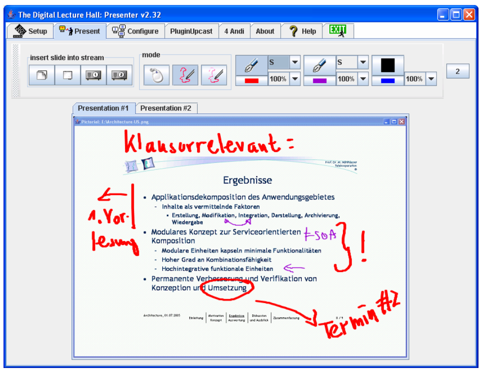


Prof. Dr. M. Mühlhäuser
Telekooperation

Ergebnisse

- Applikationsdekomposition des Anwendungsgebietes
  - Inhalte als vermittelnde Faktoren
    - Erstellung, Modifikation, Integration, Darstellung, Archivierung, Wiedergabe
- Modulares Konzept zur Serviceorientierten Komposition } ≠ SOA
  - Modulare Einheiten kapseln minimale Funktionalitäten
  - Hoher Grad an Kombinationsfähigkeit
  - Hochintegrative funktionale Einheiten
- Permanente Verbesserung und Verifikation von Konzeption und Umsetzung

Architecture, 01.07.2005 | Einleitung | Motivation Konzept | Ergebnisse Auswertung | Diskussion und Ausblick | Zusammenfassung | 8 / 9



Klausurrelevant:

Prof. Dr. M. Mühlhäuser
Telekooperation

Ergebnisse

1. Vorlesung

- Applikationsdekomposition des Anwendungsgebietes
  - Inhalte als vermittelnde Faktoren
    - Erstellung, Modifikation, Integration, Darstellung, Archivierung, Wiedergabe
- Modulares Konzept zur Serviceorientierten ≠ SOA Komposition
  - Modulare Einheiten kapseln minimale Funktionalitäten
  - Hoher Grad an Kombinationsfähigkeit
  - Hochintegrative funktionale Einheiten
- Permanente Verbesserung und Verifikation von Konzeption und Umsetzung

} !

Architecture, 01.07.2005 | Einleitung | Motivation Konzept | Ergebnisse Auswertung | Diskussion und Ausblick | Zusammenfassung | 8 / 9

Termin #2

Figure 3: Original and shrunken slide including additional annotations

system. History slides simply enlarge the sequence of slides that are currently presented to the auditorium by the usage of multiple projectors. History slides have been developed in a former project and presented in [6] that covers their realization and value.

2.1.1. Developer. Since our open architecture contains very detailed models and media-modules that cover almost the whole design and functionality of the system, the adaptation of the presented functionality was easily realized by composition and usage of very few API-commands.

To adopt the functionality, the developer just has to call the media-modules *slide-renderer*, *slide-archiver*, *slide-transformer* and *image-viewer* in the following fixed order (please see sample 1 which is implemented in Java): First of all, the system has to archive the content currently presented to the auditorium. Instead of forwarding the signals directly to our database module, we use the concepts presented in [7] to additionally send a clone of the signals to an instance of the rendering-module that basically "flattens" a slide to an image that is forwarded to the transformation-module. The transformation-module itself performs the corresponding transformation (scale and translate) and forwards the result to our image-viewer that presents the content on the presentation area, replacing the original content. For ease of understanding we decided not to use the slide-viewer in our explanations that is able to present a slide. Compared to an image-based representation, a slide consists of several layers and therefore keeps primary annotations still editable.

Compared to classroom presenter we offer the lecturer to call one of nine different transformation-matrices for this feature that places the shrunken content into the center, a corner or along a side of the rectangular presentation area. Please refer to figure 3 that contains the original slide and the shrunken, centered version including additional annotations.

2.1.2. Auditorium. Since we believe that classroom presenter's realization has a similar impact regarding the cognitive load of the auditorium, we want to present former results that prove that students feel more comfortable with history-slides. History-slides soften effects of slide-transitions and decrease the amount of students who feel "lost" and can't follow the currently presented topic any more. Several classrooms at our university contain multiple projectors that are used for projection of the current slide and up to two previously presented slides.

The add-notes sample offers a comparable intention, but also allows us to support students who attend lectures in classrooms with a single projector.

2.1.3. Lecturer. In case a common seminar room is used or the lecturer uses mobile equipment it is very likely that only a single projector is provided or can be supported by the hardware. We can support users in such environments with a function similar to our history on multiple projectors and therefore increase their conceptual adjustment and acceptability of the whole system.

Users are now able to create references to portions of the currently presented slide, even if they wish to develop big annotations. In earlier times, this was not possible, because they had to completely switch to a mode called *whiteboard-mode*. For a subset of users who primarily develop content in this mode, we offer them a version of this new feature with modified parameters (scaling is omitted and translation directed to "scroll" the content up) to increase also their benefits: Slides will not be shrunken, but scrolled up so that portions of formerly created content will disappear smoothly and are replaced by a blank annotation area that appears at the bottom.

Adopting classrooms presenter's feature and extending it for the special support of our whiteboard mode allows us to emulate some of the principles of an overhead projector. In the case of scrolling our concept is getting more similar to digital sliding boards like E-Chalk [8]. A difference we may introduce to our work is E-Chalk's possibility to display content that has just been scrolled out onto a succeeding display.

2.1.4. Supporters. The hardware for the single projector solution is much easier to assemble and often allows keeping existing systems unchanged.

Nowadays, common notebooks, convertibles and desktop systems contain a secondary output for video signals. In contrast, a system for our multiple projector solution is typically built on purpose and contains either two PCI Express cards that require a special motherboard or a quad graphics adapter from Matrox Graphics Inc. [9].

2.2. Integrate self-contained/external systems

Our architecture offers several extension points which can be used by developers of external applications to integrate their application respectively functionality. Our highly extensible system serves in that case as a core system that will be extended to meet the individual requirements of different users and universities who want to use a specialized version. In the given example, we present the integration of components that support synchronous communication within a lecture.

2.2.1. Developer. To assist developers in several other aspects, we deliver a comprehensive software development kit that contains the documented API, a tutorial that explains the sample development of extending components, several precompiled samples including source-code and a testing-environment that is displayed in figure 4. The latter allows rapid integration of components, basic testing and use of API-functions without the overhead of a fully installed whole system.

To integrate components, it is necessary to fulfill several contracts (basically a set of interfaces). Besides meta-data like *author's name* and *description of the functionality*, it is possible to extend the user-interface of the lecturer by providing him with additional contexts and ambient displays that are currently visualized as panels and tabs within a tabbed pane (see figure 3). Optional contracts allow components to register for dedicated events, to be notified or to publish events themselves. The core system fires events when the user performs a context change. In case a specific context like the one that handles incoming questions of the students has been selected it is therefore possible to reset a counter on the projected area that is visible for the auditorium. It allows notifying the students in parallel to the lecture that the lecturer has scanned their questions. An event that is always fired when the interactive components receive an incoming question forces the core system to update the ambient displays, which is basically the corresponding tab of a context-panel. The lecturer will be provided with dynamic icons or strings that represent the amount of incoming questions that he has not scanned yet. It is also possible to fire an event that automatically invokes a switch to its parent context.

Systems that do not offer or currently use this kind of coupling of presentation and interaction are the presentation and recording system Lecturnity [10] and Wil-Ma [11] that supports interaction in the classroom.

2.2.2. Auditorium. The auditorium is able to follow the lecture smoothly, since the lecturer does not switch between different applications or even systems. The framework also supports the synchronized display of content or questions submitted by students. Content that will be presented / removed by the method *publish(Slide s)* / *unpublish()* automatically becomes part of the recording. For later retrieval, the corresponding slide-model contains in addition to temporal information the description of the content; here, the type is *submission by student*.

2.2.3. Lecturer. The integration of other systems allows the lecturer to access a wider range of educational and pedagogical methods. An interactive lecture is created by integration of communicational components as additional contexts. The view-mechanism of the open architecture assists asynchronous and synchronous communication during the lecture. The user doesn't have to switch circuitously between several applications or even systems, but works confidently in a specific context. The handling of several contexts is arranged clearly and assisted by the system in different ways, e.g., an important change in the model of a context can force a notification (currently a visual signal) of the lecturer.

Our architecture provides system wide access and manipulation of semantic information, for example it is possible to assign questions of students to a specific presented slide. In case the lecturer decides to communicate the answer to this question, the corresponding slide can be automatically represented.

2.2.4. Supporters. One of various advantages for supporters is our solution for the alignment of multiple areas on multiple display devices.

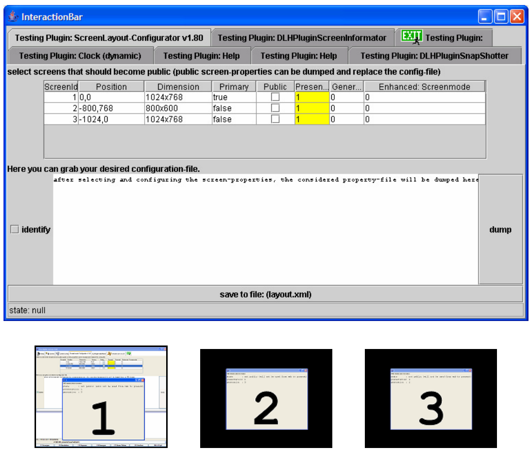
Instead of placing the different user-interfaces and display areas (remember the interaction-counter that is visible for the auditorium) of several independent systems each time, the supporter simply performs a teach-in using our module. In figure 4 you can see the user-interface for this module, since its context is currently focused. It allows configuration of each presentation area and each plugged component. Usually the given teach-in has to be performed once in a semester or only when the user decides to add additional components within the semester. Also dialogs for e.g. identification are bundled and clearly presented once.

2.3. Integrate into external environments

In this section, we focus on the integration into environments that already support face-to-face learning by usage of several external components. Based on our open architecture, we are e.g. able to integrate the presentation part of our system into existing recording-environments.

2.3.1. Developer. Existing and universal synchronization mechanisms allow to easily support such recording-environments and to dramatically reduce the amount of written code.

Instead of replacing an external environment that performs recording and likely additional tasks we establish a simple network connection between both environments. A recording server has to provide the basic functions *begin/end recording* and *mute/enable microphone*. In addition a feature called *change recording* can be provided to support SMIL-based replay of recordings that rely on audio snippets, each matching a snapshot of the slide that has been displayed and stored on-the-fly by the presentation system. This technique requires a media player that copes with incremental replay of files without noticeable gaps.


InteractionBar

select screens that should become public (public screen-properties can be dumped and replace the config-file)

ScreenId	Position	Dimension	Primary	Public	Presen...	Gener...	Enhanced: Screenmode
1	0,0	1024x768	true	☐	1	0	0
2	-800,768	800x600	false	☐	1	0	0
3	-1024,0	1024x768	false	☐	1	0	0

Here you can grab your desired configuration-file.

```
after selecting and configuring the screen-properties, the considered property-file will be dumped here
```

state: null

Testing Plugin: ScreenLayout-Configurator v1.80
Testing Plugin: DLHPluginScreenInformator
EXIT
Testing Plugin:
Testing Plugin: Clock (dynamic)
Testing Plugin: Help
Testing Plugin: Help
Testing Plugin: DLHPluginSnapShotter
identify
dump
save to file: (layout.xml)
1
2
3

Figure 4: Teach-in of multi-projector-settings

We concentrated on the development of a small component that performs basic operations regarding the connection-handling (e.g. establishment or error handling) and remotely controls external systems. In this case, we had to use three extension points: *app.state-extension*, *slideEvent-extension* and one of the *user-interface extensions* (e.g. a view that provides a complete context or contributes to the toolbar). The *user-interface extension* can be used to provide the lecturer with a toggle button that mutes/enables the microphone and becomes part of the toolbar. The app.state-extension coordinates the beginning and ending of the recording on the remote system. The begin signal will be distributed as soon as our core-system has been successfully initialized and enabled for user inputs. The shutdown process of our presentation system includes the notification of all components that subscribed for the app.state-events, so that other components can independently perform their own and specific shutdown procedure.

The slide-event is generated each time a slide-change is performed or other media is integrated via the framework. For example, turning into whiteboard-mode, displaying a student call-in or using the previously presented feature add-notes perform such an event that usually results in archiving the content and creation of a new audio file. Therefore, our component subscribes to latter event and sends the command *change recording* to the recording environment whenever it is triggered by the framework.

2.3.2. Auditorium. The auditorium is provided with a recording that allows recapitulating the lecture and is trimmed according the lifecycle of the presentation system. Indices are created during the presentation in an intelligent and not manual way, without distracting the students. They automatically receive meaningful navigational indices for their individual playback that is greatly assisted by full-text search over extracted slide content. Additional content like the extracted slide titles/content and the description of the course (e.g. date, lecturer) can be used for semantic retrieval.

2.3.3. Lecturer. Since the drawings of the lecturer are performed on an interactive display and integrated into the recordings, he can conceptually rely on this feature and reuse hand drawings in upcoming talks, even without having manually controlled the recording.

2.3.4. Supporters. Since appropriate timings and cuttings are automatically generated according to emitted events of the presentation system, this eliminates also the computational detection of synchronization indices like developed in [12] and decreases the timing problems presented in [13]. Supporters only need to take care about a reliable network connection between both systems and manually reedit the recordings in case a given sequence should not be published, besides the fact that the lecturer is able to use a simple toggle button that mutes/enables the microphone, so he can suppress unwanted audio already during the lecture.

For automatic distribution of the content, the system described in [14] can be used. It mainly uses a peer-to-peer approach to assist mobile learners in this context.

3. Summary of contributions

We presented an open architecture for face-to-face learning that relies on our concept of categories and highly reusable media-modules. To illustrate the manifold values we selected examples that demonstrate its extensibility and interoperability. In particular, we described how to:

- extend presentation facilities
- integrate self-contained/external systems
- integrate into external environments

In addition, we enumerated the advantages and values for developers, lecturers, learners and supporters. We have shown that our solution is highly extensible, open to integrate external components / systems and can even be integrated into external systems and heterogeneous environments.

4. Outlook

If we consider our whole development and current state (please refer to figure 1), we believe that the definition of a standard for interchangeable components in the context of face-to-face learning should be considered for future work.

We already have experiences from development and usage of several compatibility layers within our own system that allow us to smoothly migrate between different versions of the core system and the underlying API. It would be interesting to use such mechanism to create compatibility between different systems and move towards a uniform standard.

Since we believe that working on such a standard would be a major contribution to the community, we invite you for any cooperation.

5. References

[1] Georg Turban, Max Mühlhäuser, “A category based concept for rapid development of ink-aware systems for computer-assisted education,” In: Proceedings of the 7th IEEE International Symposium on Multimedia, Irvine, California, USA, pp. 449-457, 2005.

[2] Joshua Bloch, Google, Inc., USA, “How To Design a Good API and Why it Matters,” keynote on Library-Centric Software Design at Object-Oriented Programming, Systems, Languages and Applications conference, San Diego, California, USA, 2005.

[3] Robert Seacord, Lutz Wrage, “Replaceable Components and the Service Provider Interface,” Carnegie Mellon University, Software Engineering Institute. http:// www.sei.cmu.edu/publications/documents/02.reports/02 tn009.html, September 22nd, 2006.

[4] University of Washington, “UW Classroom Presenter,” http://www.cs.washington.edu/education/dl/presenter/, September 22nd, 2006.

[5] Richard Anderson, Ruth Anderson, Luke McDowell and Beth Simon, “Use of Classroom Presenter in Engineering Courses,” In: Frontiers in Education, 2005.

[6] Max Mühlhäuser, Christoph Trompler, “Digital Lecture Halls Keep Teachers in the Mood and Learners in the Loop,” In: Proceedings of E-Learn 2002, Montreal, Canada. pp. 714-721, Association for the Advancement of Computers in Education, Charlottesville, VA, 2002.

[7] Stefan Hirtbach, “Pelagia: Ein Multimedia-Framework - Konzept und Implementierung” diploma thesis, Darmstadt University of Technology, Darmstadt, Germany, 2006.

[8] Freie Universität Berlin, Institute of Computer Science, “Projekt E-Chalk,” http://www.echalk.de, September 22nd, 2006.

[9] Matrox Graphics Inc., “Matrox G450 MMS (quad-display graphics card),” http://www.matrox.com/ graphics/pid/products/mms/G450x4_mms.cfm, September 22nd, 2006.

[10] Imc advanced learning solutions, “Lecturnity,” http://www.lecturnity.de/, September 22nd, 2006.

[11] University of Mannheim, “Wireless Interactive Lectures in Mannheim,” http://www.lecturelab.de/, September 22nd, 2006.

[12] Peter Ziewer, “Navigational Indices and Full Text Search by Automated Analyses of Screen Recorded Data,” In: Proceedings of E-Learn 2004, Washington, DC, USA, 2004.

[13] Sugata Mukhopadhyay, Brian Smith, “Passive capture and structuring of lectures,” ACM Multimedia, p. 477-487, 1999.

[14] Marc Dillmann, “Intelligent Satchel: Entwurf und Implementierung eines Systems zur automatischen Verteilung und Synchronisation von vorlesungsbezogenen Inhalten und Informationen,” diploma thesis, Darmstadt University of Technology, Darmstadt, Germany, 2006.