An Integrated System for Interaction Support in Lectures

Henning Bär
hcbaer@rbg.informatik.tu-darmstadt.de

Gina Häussge
huge@rbg.informatik.tu-darmstadt.de

Guido Rößling
guido@rbg.informatik.tu-darmstadt.de

Department of Computer Science
Darmstadt University of Technology
64289 Darmstadt, Germany

ABSTRACT

This paper motivates the use of an interaction support system both during and outside lectures. We describe requirements for an "optimal" system based on an investigation of related interaction support systems. An example system architecture that satisfies these requirements is presented. The evaluation of the system clearly shows its usability.

Categories and Subject Descriptors

K.3.1 [**Computers and Education**]: Computer Uses in Education—*Computer-assisted instruction*

General Terms

Design

Keywords

Lecture interaction, Architecture, Moodle, OpenOffice.org

1. INTRODUCTION

Interaction is an essential part of the learning process, both the interaction between students and between students and educators. However, interaction in large lecture venues with more than 100 students is at best difficult to achieve. It becomes impossible if the number of students exceeds the capacity of the lecture hall, so that video transmission is used to reach all students.

But even in small lectures attention can be increased. According to Lloyd [7], the attention drops significantly after only 20 minutes. Ruhl and Suritsky [10] reached better learning results by pausing for only 2 minutes after each lecture period of 20 minutes. Even though their students had learning disabilities, the same effect may apply for other students. A pause is a media break similar to other breaks, such as shifting from slides to discussion.

Section 2 describes some related interaction support systems, which use a media break from teaching the students to quizzing them. Based on these systems, we have extracted a set of requirements for an "ideal" interaction support system. Section 3 explains the structure of such a system we developed and how it addresses the requirements. Some properties and features are highlighted, which we think are unique among interaction support systems. Section 4 summarizes evaluation results of the use of our system.

ITiCSE'07, June 23–27, 2007, Dundee, Scotland, United Kingdom.

2. INTERACTION SUPPORT SYSTEMS

There are many different systems which aim at an increase of interaction. They can be organized in different categories. One group of systems are Learning Management Systems (LMS). The rest can be distinguished by the way students use the interaction. They can use it in a browser, as an application on a notebook, on a Personal Digital Assistant (PDA), on a mobile phone, or with special devices. Of course, combinations are also possible within one system. We will present one representative system for each type.

Learning Management Systems.

Learning Management Systems are typically used with a web browser outside the lecture. The educator can upload course material and create quizzes. Interaction can take place using shared whiteboards, chats, e-mail, and forums. One LMS server can host several lectures. To distinguish the users, the educator and the students have to authenticate themselves. The authenticated users have to be inscribed in a course to order to use it.

Moodle [3] handles most components as plug-ins, allowing them to be easily replaced. Additional plug-ins can be created. External systems such as a Lightweight Directory Access Protocol server (LDAP) [11] can be used for authentication to integrate Moodle into an existing infrastructure. Moodle user roles can also be acquired from LDAP.

Browser-based interaction.

Systems based on browsers can easily support interaction during lectures. Students navigate to a web site, through which they can interact with the educator.

One such system is *ClassInHand* [2]. The educator uses a PDA to host a web server. The students can connect using a browser and type in questions about the content of the lecture. They can also rank the educator by choosing a number between -10 and +10. The educator can ask students multiple choice questions and retrieve an aggregation of their answers. The students do not have to authenticate for the interaction, only for downloads or other web server-related features offered by the educator.

Notebook-based interaction.

Most interaction systems that can be used with a notebook can also be used with other devices or in a browser. A notebook takes up almost the entire table space a student usually has. A survey in 2006 of computer science students showed that 78% preferred notebooks over mobile phones (12%) or PDAs (8%). In computer science courses, students are also well-equipped with notebooks: 86% owned a notebook, 14% a PDA, and 80% a mobile phone.

Lectcomm [4] consists of a server application with integrated functionality for the educator, including an add-in for PowerPoint. Additionally, there is a client application for the students, which can be started as a standalone application or as an applet. When the educator posts a quiz, the students' clients automatically open the question or exercise. No authentication is required. Lectcomm supports only one lecture.

PDA-based interaction.

Many systems concentrate on PDAs. Such systems do not differ much from those used on notebooks, since typical PDAs also offer wireless LAN. An advantage is that PDAs are far smaller, so students can still take notes. However, the text input can be somewhat cumbersome.

The *Wireless Digital Learning Assistant* [6] offers PDA-based interaction. Students can submit questions and other text messages to the educator. The educator can post information about the lecture as well as quizzes. To interact during a lecture, students have to authenticate. Their accounts are created by the educator. Students can review their own messages or the quizzes of the lecture.

Mobile phone-based interaction.

Current mobile phones are powerful enough to be used for interaction support. They can communicate via GSM, many are equipped with Bluetooth, and some even offer wireless LAN. Most students have their own mobile phone and are used to the rather cumbersome input possibilities.

Information Communication Technology [8] uses the SMS functionality of mobile phones. The system anonymizes the message for the educator but stores the sender information internally. To access their messages afterwards with a browser, the students have to authenticate.

Interaction based on special devices.

Some commercial systems consist of an infrared or radio receiver connected to the educator's notebook. Quiz questions can be displayed using a projector, so all students can see them. They can then answer the question using remote controls which were given to them.

Digivote [5] offers 10 keys on each remote control. Some of the keys are colored, so they can match the colors of possible answers on the projector. Each remote control can be associated with a student to perform personalized quizzes. Due to the lack of a keyboard or other input method, students cannot formulate specific questions.

2.1 Fundamental Requirements

An ideal interaction supporting system has to fulfill a set of requirements. For the sake of conciseness, we focus here on three meta-goals:

1. Notebooks are the most convenient device for interactions in class partially because they can also be used for accessing the LMS. Among other reasons, this led to the request that the system should support interaction during the lecture as well as outside of the lecture, as supported by *Moodle*, the *Wireless Digital Learning Assistant*, and *Information Communication Technology*.

2. An open system architecture can address the incompatibility problems between the (many) existing systems. It should also support future extensions and different user devices. Moodle offers a plug-in-based framework to integrate new functionality. However, a way to support notebooks, PDAs, mobile phones, and special devices has to be found.

   The goal of our research differs in this key aspect from other approaches. We want to specify an open, unified architecture to be used as a basis for interactions systems. Many other related projects focus only on a specific interaction system.

3. The interaction system has to be scalable from test installations over use in one lecture to an installation for an entire campus. For short tests, *Lectcomm* [4] offers a server which can be started immediately after downloading. Moodle solves the installation for one and many lectures.

One can think of many further requirements. Due to the required open architecture, most other requirements can be achieved in a second step. However, none of the evaluated systems (far more than could be listed here) supported all required features. We therefore started to develop an open system architecture that would be able to fulfill all requirements and be easy to adapt by other users. This system will be described in the next sections.

3. SYSTEM DESCRIPTION

We designed an interaction support system in three layers, as shown in Figure 1. The first layer acts as a tool chain, which mainly handles the persistence of objects. We used the Hibernate library [9], which allows us to switch between one out of 22 database backends by simply modifying one configuration file. The layer also offers further functionality, such as sending e-mails or authenticating users. It is not aware of any other part of the system.

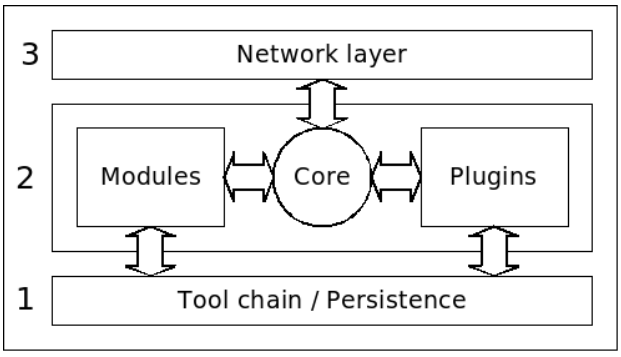

Figure 1: Structure of the developed system

The middle layer is called program logic and does the main work. It handles the requests using the tool chain layer. To offer a simple interface which can process complex data, a generic method processes objects of type *EmptyRequest*. This type of request essentially consists of some authentication data and a reference to the according lecture. Of course, inherited objects can also be passed to this method.

The third layer is responsible for the transfer of the data from the interaction clients to the server. We are currently using web services for compatibility reasons.

To keep the system extendable, it is also split into plug-ins and modules orthogonal to the layers. Plug-ins and modules are managed by a central core component. This core is aware of registered plug-ins and modules and decides whether to accept incoming requests or not. Whereas each plug-in can be removed, all modules are mandatory. However, even modules can be replaced by a different implementation of the module's functionality.

One such functionality realized by a module is the authentication. For testing purposes, we provide an implementation that performs no password validation, so that any user can log in. For campus use, this implementation can be replaced by one which makes use of an external authentication infrastructure. Another functionality provided by a module is the logging subsystem.

The types of interaction are handled by plug-ins. The five plug-ins we developed allow students to answer multiple choice quizzes, to evaluate parameters of the lecture, to submit text messages such as questions, to retrieve textual information, and to retrieve the current slide.

3.1 Student clients for use during lectures

We were able to offer the use of the interaction on notebooks, PDAs, and mobile phones. Figure 2 shows that the mobile phone client consists of a simple menu and an interface for each interaction type. Before using the application, students have to authenticate. By doing so they receive a random text, called *token*, which is also stored on the server. The next times the application is started, the token can still be used and the user does not have to type in username and password again.

Figure 2: Student client for mobile phones

On notebooks and some PDAs, the student client can show all interaction elements at once to achieve a minimally distracting application. Depending on the installed Java virtual machine, the application on the PDA can also be identical to the one for mobile phones.

In a survey in 2004, we asked the students whether they would pay the GSM fee for connecting with their mobile phones. Only 4 out of 72 were willing to do so. To offer a free of charge solution, we installed a Bluetooth access point in one of our lecture halls and extended the client application for Bluetooth use [1].

3.2 Educator client for use during lectures

The educator uses a client which is divided into two parts: one for student-initiated content (such as questions and evaluation), and the other for educator-initiated content, especially quizzes. The division in two parts separates the logically different elements. Additionally, both tabs are less crowded and easier to use by the separation between incoming feedback and responses to prepared quiz questions. Figure 3 shows them as tabs in a tabbed pane.

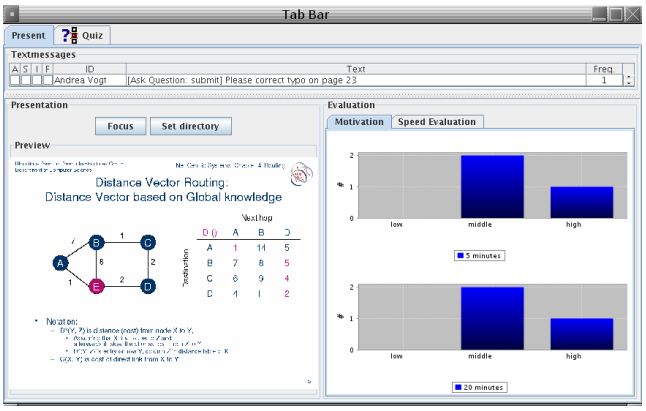

Figure 3: Educator client for in-lecture interaction

The activated tab shows the students' text messages at the top. The educator can choose a message, show it on the projector and discuss it with the class. The presentation is shown below on the left. A control of a presentation software has to implement a certain interface to be integrated. We used the OpenOffice API, which does not support buttons for backwards or forwards. Instead, the educator can click on the "Focus" button shown in Figure 3 and use the software as usual. The bar diagram on the right shows the students' ranking. Thus, the educator can see all interaction information including a preview of the presentation at a glimpse.

The quiz tab (not activated in Figure 3) is used to ask the students quiz questions. The question and the possible answers will be shown on a projector, so even students without mobile devices can think about the question.

A "co-pilot" feature can allow colleagues - usually, assistants of the educator - to screen incoming questions. They can then answer some of the questions directly by sending a reply by e-mail. Questions that the assistant does not feel able to answer, or that he believes to be of interest to the wider audience, are forwarded to the educator by the "co-pilot". This features reduces the workload of the educator by accounting for large parts of the incoming ("easy") questions.

3.3 Other Uses

According to the requirements, the interaction system also has to support access outside the lecture. Since we use web services for the communication, we could create a PHP application to offer access for educators and students. Figu-

re 4 shows that we integrated it as a plug-in into the Moodle LMS, so students and educators do not have to find another web site or become familiar with a different tool.

Figure 4 shows an interface where students can choose in a tab-based view between the offered interaction elements laid out in separate tabs. The selected tab shows the text messages. There they can find their own messages and possible answers, as well as text messages that were displayed during class. Their questions are possibly related to content they did not understand. They can use the interface to check before the examination whether their questions are still open. In other tabs, students can recall the evaluations or train themselves using the multiple choice quizzes. The educator can also see the interaction messages of the lecture or use the interface to create multiple choice quizzes.

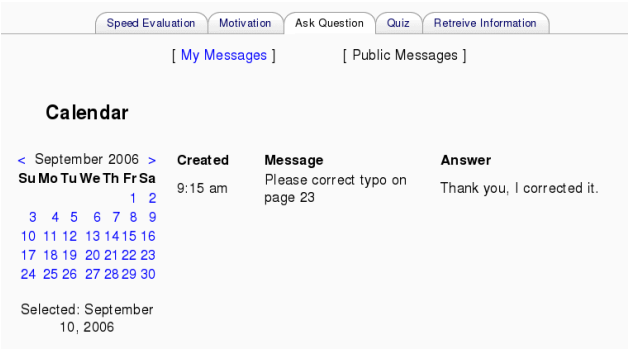

Figure 4: Client for use in the LMS Moodle

We had to decide how to deal with the issue of anonymous use by students. Students have to be authenticated to review their own messages after the lecture. In the poll in 2006, we asked them how they believe the usage of the interaction system would change if their identity were no longer hidden. Only 2 students assumed the usage would increase, 18 students believed the change would not have any effect, but 47 students thought the usage would decrease.

We therefore chose to automatically create pseudonyms for the users. Each pseudonym can be used as an e-mail address. A resolving server can accept incoming mail, remove the mail headers, change the sender address to the associated pseudonym, change the recipient's pseudonym to the real address, and forward the e-mail. By this procedure, students can even communicate with the educator using e-mail without losing their anonymity. Additionally, an educator can see whether all questions are from the same student or from different students.

4. EVALUATION

How well does our system match to the requirements in Section 2.1?

Adaptability: The web services we chose as a protocol allowed us to use the interaction system with Java clients on notebooks, PDAs, and mobile phones. We also successfully connected .NET clients for PDAs and notebooks to offer Microsoft Windows users a seamless look-and-feel. We reduced the required communications for each client to a minimum. Thus, a student was able to develop a PHP-based student client within one afternoon.

The server application allows the replacement of layers. Switching the communication to Remote Method Invocation (RMI) required less than 100 lines of code. The persistence library Hibernate [9] supports 22 database backends, selected by editing a configuration file.

Recall: Clients were developed for minimal distraction during lecture for students as well as for educators. A plug-in for the Moodle LMS was created to access the interaction data outside of lectures, such as repeating multiple choice quizzes. Additionally, our approach allows us to process interactions by e-mail without losing the students' anonymity.

Scalability: The server application is designed as a servlet. To run it, it only has to be copied into a servlet container and the container has to be started. For a test scenario, such as an exhibition, the storage can run in a file-based database, so no further installation is required. A scalable database system can be used to serve an entire campus.

We also provide an authentication module for performing quick tests. The module distinguishes user login names without verifying the password. For campus scenarios, another authentication module can forward the authentication requests to external systems used on campus. The same applies to the handling of lectures. It is possible to use a module which declares any authenticated user automatically as a participant of the lecture. For campus use, the system can request the lectures a student attends from external systems, for example from the LMS Moodle.

The system must be sufficiently fast to be able to act as a server for an entire campus and to host several lectures at the same time. In our most active lecture, a new text message arrived on average every 50 seconds. If 50 such lectures took place in parallel, the message rate could rise to $1/s$. To be able to handle load peaks, we tested the system with 10 submissions per second.

The server was installed on a Celeron 1000 Laptop and for a time period of 10 minutes, random text messages from 10 different computers were sent. Figure 5 shows the load of the server as measured while the messages were processed. Unix systems use the *load* as a measure of how many processes are queued by the operating system to be processed. When starting up the Integrated Development Environment *Eclipse*, the load easily reaches values beyond 2.0. The peak value of 0.6 in Figure 5 on a comparatively slow machine therefore shows that the server can easily handle the expected number of submissions.

We used the application in several different lectures. The lectures were evaluated at the end to determine what our students thought of the interaction support system. In the following, we present results from these evaluations.

The application must not be too difficult to install. We decided to use Java WebStart for the distribution process, so the students had to do three things before the application started: they had to open a browser, type in a URL which the educator announced, and then had to click on a link. We

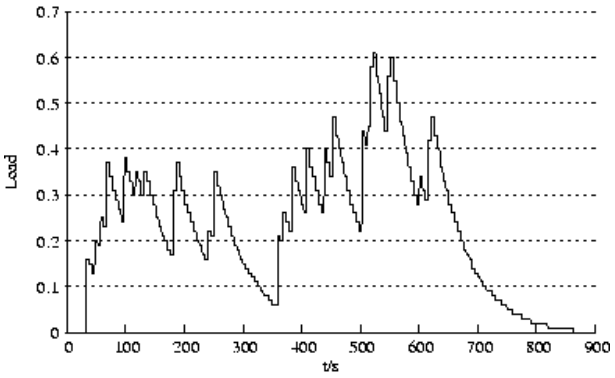

Figure 5: Server "load" for 10 messages per second

asked the students how difficult the installation process or the steps before the start were. The mean value was 3.6 on a scale from 1 (very difficult) to 5 (very easy). A closer investigation shows that a few students had technical problems; without them, the mean result was 3.8.

It is also important that the application is sufficiently simple to use. On a scale from 1 (very difficult) to 5 (very easy), the mean difficulty rating by 49 students in the evaluation in 2006 was 4.1, including those students who had technical problems. Thus, using the application caused no significant problems even for those who had trouble getting it installed.

We also asked the students if their involvement with the subject was increased by the multiple choice quizzes. On a scale of 1 (not at all) to 5 (very much), the mean value of 2.9 was less than expected. However, one educator pointed out that it can be difficult to create meaningful questions. The correct answer should not be obvious, but clear reasons are needed why the other answers are incorrect. Additionally, we assume that the students' involvement could increase if the educator took the time to discuss the wrong answers chosen by at least some students with the class.

After the introduction of a free-of-charge solution for mobile phones, we asked the students in 2006 which device they used for the system. For mobile phones, they also had to answer whether they communicated by GSM or by Bluetooth. Out of 46 participants, 14 used a mobile phone. Nine of them used Bluetooth and only two students used GSM. The other three students communicated with both.

5. SUMMARY AND OUTLOOK

We have presented a system which can support interaction between students and educators during the lecture as well as outside the lecture. It has an open architecture, all interaction types are loaded as plug-ins and the replacement of layers was tested. As a servlet, the system is scalable from small test scenarios to a large campus.

Further interaction extensions based on this framework can improve the learning process of students. One extension which is almost finished is based on our lecture recordings. Using Flash we can provide a combination of slides, video, and audio to the students. Flash also allows access to web services, so the students will be able to access their own interactions synchronized with other recordings of the lecture.

6. ACKNOWLEDGMENTS

We thank CellIQ for lending us different mobile phones for our research. Siemens Mobile also gave us two brand new Bluetooth-capable mobile phones.

7. REFERENCES

[1] H. Bär, G. Rößling, E. Tews, and E. Lecher. Bluetooth Interaction Support in Lectures. In *Proceedings of Mobile Learning 2006*, pages 360–364. IADIS Press, 2006.

[2] A. L. Bishop, R. K. Dinkins, and J. L. Dominick. Programming handheld devices to enhance learning. `http://www.educause.edu/ir/library/pdf/EQM0318.pdf` (seen March 27, 2007), 2003.

[3] M. Dougiamas and P. C. Taylor. Moodle: Using Learning Communities to Create an Open Source Course Management System. In *Proceedings of the World Conference on Educational Multimedia Hypermedia & Telecommunication (ED-MEDIA), Honolulu, Hawaii, USA*, pages 171–178. AACE Press, Charlottesville, VA, 2003.

[4] T. Gross, L. Szekrenyes, and C. Tuduce. Increasing Student Participation in a Networked Classroom. In *Proceedings of Frontiers in Education*, pages 9–13. IEEE Press, 2003.

[5] P. E. Jones. Improving learning in lectures using keypad-response units. In *The Proceedings of the 8th Annual Teaching Learning Forum*, pages 173–177, 1999.

[6] T.-C. Liu, H.-Y. Wang, J.-K. Liang, R.-W. Chan, and J.-C. Yang. Applying Wireless Technologies to Build a Highly Interactive Learning Environment. In *Proceedings of the IEEE International Workshop on Wireless and Mobile Technologies in Education (WMTE'02)*, pages 63–70. IEEE Computer Society Press, Los Alamitos, CA, USA, 2002.

[7] D. M. Lloyd. A concept of improvement of learning response in the taught lesson. *Visual Education*, pages 23–25, 1968.

[8] C. Markett, I. A. Sánchez, S. Weber, and B. Tangney. "Pls Turn ur Mobile on": Short Message Service (SMS) Supporing Interactivity in the Classroom. In *Proceedings of the IADIS International Conference Cognition and Exploratory Learning in Digital Age (CELDA 2004)*, pages 475–478. IADIS Press, 2004.

[9] Red Hat Middleware. Hibernate website. `http://hibernate.org` (seen March 27, 2007).

[10] K. Ruhl and S. Suritsky. The Pause Procedure and/or Outline: Effect on Immediate Free Recall and Lecture Notes Taken by College Students with Learning Disabilities. In *Learning Disability Quarterly, 18*, pages 2–11, 1995.

[11] M. Wahl, T. Howes, and S. Kille. Lightweight Directory Access Protocol (v3). `http://www.ietf.org/rfc/rfc2251.txt` (seen March 27, 2007), 1997.