The Interactive Lecture: A new Teaching Paradigm based on Pervasive Computing

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von
Diplom-Wirtsch.-Inf. Nicolai K. Scheele
aus
Berlin

Mannheim, 2005

Dekan:	Professor Dr. Matthias Krause, Universität Mannheim
Referent:	Professor Dr. Wolfgang Effelsberg, Universität Mannheim
Korreferent:	Professor Dr. Colin Atkinson, Universität Mannheim

Tag der mündlichen Prüfung: 17. Januar 2006

Abstract

Lectures, though often criticised for their monolithic instruction style and - associated with that - the lack of motivation on the side of the students leading to a very low learning success, are still one of the most efficient educational methods known in higher education. In addition, lectures are very adaptive to time tables, other courses, different audiences and new cognitions, and they play a valuable part in the social life of the students.

In this dissertation, a novel approach to overcome the deficits of lectures is presented. With electronic means the students are able to give feedback, ask questions or take part in small knowledge tests during the lecture; the lecturer, supported by automatic aggregation and analysis can immediately respond to the information received on this additional communication channel. This way, the students are more actively involved in the lecture and thus retain a higher motivation.

The software system specifically designed for the "interactive lecture" is explained in detail: "WIL/MA" (Wireless Interactive Lectures in Mannheim) contains all components needed to set up the required configuration fast and efficiently. Using only mobile and light-weight devices with wireless connectivity, no special preparation

of lecture halls is needed; neither are the students harassed with clumsy computers occluding the view to the teacher.

As a second major part of this dissertation, five extensive experiments, performed to investigate the effects and problems of the new scenario, are discussed along with the results that could be deduced. Furthermore, the WIL/MA tools have been used frequently in courses of two different faculties over the last three years, hence, based on our experience, we were able to derive valuable advice for an efficient application.

Zusammenfassung

Vorlesungen sind wegen ihres einseitigen Präsentationsstils und den damit verbundenen motivationalen Problemen, die zu einer geringeren Lernleistung führen, seit langer Zeit kritisiert worden. Dennoch gelten sie als eine der effizientesten Lehrmethoden an Hochschulen, die eine Vermittlung von Wissen an eine große Zuhörerzahl gleichzeitig gestatten. Ausserdem besitzen sie auch eine hohe Adaptivität bezüglich zeitlicher Restriktionen, anderen Kursen, verschiedenen Hörerkreisen und neuen Erkenntissen; sie sind ferner ein wichtiger Bestandteil im sozialen Alltag der Studenten.

Diese Dissertation beschreibt einen innovativen Ansatz, die Defizite der Vorlesung zu beseitigen. Mit elektronischen Mitteln können Studenten während der Vorlesung Rückmeldungen senden, Fragen stellen oder an Wissenstests teilnehmen. Der Dozent kann mit Hilfe einer automatischer Analyse und Zusammenfassung der eingehenden Informationen direkt und unmittelbar auf diese eingehen. Studenten werden auf diese Weise aktiver in das Unterrichtsgeschehen eingebunden und dadurch stärker zur Teilnahme motiviert.

Das Softwaresystem “WIL/MA”, das speziell für die Anwendung in der “interaktiven Vorlesung” implementiert wurde, wird ausführlich beschrieben. Es enthält alle

Bestandteile, die zum schnellen und effizienten Aufbau der benötigten Infrastruktur erforderlich sind. Da nur leichte und mobile Geräte in diesem Szenario verwendet werden, ist keine spezielle Vorbereitung der Hörsäle vonnöten; ebenso werden die Studenten nicht durch große, die Sicht versperrende Computer in der Vorlesung gestört.

Ein zweiter, großer Teil der Dissertation widment sich fünf umfangreichen Untersuchungen, die durchgeführt wurden um das neue Szenario auf positive Effekte oder Probleme hin zu untersuchen. Diese werden mitsamt der sich daraus ergebenden Ergebnisse detailliert offengelegt. Weiterhin wurde die WIL/MA Architektur regelmäßig in Kursen zweier Fakultäten eingesetzt, so dass wir auf Grund unserer Erfahrungen wertvolle Hinweise zum effizienten Einsatz geben können.

Danksagungen

Die vorliegende Arbeit entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Praktische Informatik IV der Universität Mannheim. Der überwiegende Teil dieser Arbeit wurde in dem vom BMBF geförderten Verbundprojekt VirOR (Virtuelle Hochschule Oberrhein), sowie dem von der DFG finanzierten Projekt "LectureLab" durchgeführt. Sowohl dem BMBF als auch der DFG möchte ich für die finanzielle Unterstützung hiermit danken.

Ein besonderer Dank gilt dem Betreuer dieser Arbeit, Herrn Professor Dr. Wolfgang Effelsberg. Er gab mir die Möglichkeit zur freien wissenschaftlichen Entfaltung, zögerte aber nie, mir in zahllosen konstruktiven Diskussionen unterstützend zur Seite zu stehen. Mit viel Engagement und Interesse verfolgte er den Verlauf des Projektes und gab mir mehrfach die Möglichkeit, die im Rahmen der Arbeit entwickelte Software in seinen Vorlesungen einzusetzen. Darüberhinaus ermöglichte er mir die Teilnahme an internationalen Konferenzen, sowie einen mehrmonatigen Aufenthalt am Stanford Center for Innovations in Learning der Stanford University.

Ebenfalls möchte ich mich bei meinem Zweitgutachter, Herrn Professor Dr. Colin Atkinson und bei dem Dekan der Fakulät, Herrn Professor Dr. Peter Krause be-

danken; weiterhin bei meinen Kolleginnen und Kollegen Marcel Busse, Ursula Eckle, Dirk Farin, Stefan Fries, Holger Füßler, Thomas Haenselmann, Betty Haire-Weyerer, Volker Hilt, Holger Horz, Thomas King, Stephan Kopf, Christoph Kuhmünch, Gerald Kühne, Fleming Lampi, Christian Liebig, Tanja Mangold, Michael Möske, Walter Müller, Claudia Schremmer, Moritz Steiner, Matthias Transier und Jürgen Vogel für die angenehme, stets hilfsbereite Zusammenarbeit und die vielen Diskussionen. Besonders möchte ich mich auch bei Herrn Professor Dr. Martin Mauve bedanken, der mich ermutigt hat, den Weg zur Promotion einzuschlagen.

Bei der Durchführung des Projektes hatte ich das große Glück, eng mit dem Lehrstuhl für Erziehungswissenschaften II der Universität Mannheim unter der Leitung von Herrn Professor Dr. Manfred Hofer kooperieren zu können. Ihm danke ich für seine Unterstützung und die Möglichkeit, die entwickelte Technik in einer geisteswissenschaftlichen Vorlesung einsetzen zu dürfen. Ein besonderer Dank gebührt meiner Projektpartnerin Anja Wessels für die hervorragende Arbeit bei der Planung, Durchführung und Evaluierung unserer Experimente und für ihre Geduld, mir die Grundlagen aus der Didaktik und Statistik näher zu bringen; vor allem aber für die langjährige, freundschaftliche Zusammenarbeit an die ich mich immer gerne erinnern werde.

Nicht zuletzt möchte ich mich aber auch herzlich bei meinen Eltern, Hans-Joachim und Brigitte Scheele bedanken, die mir nicht nur das Studium und die Promotion ermöglicht haben, sondern auch jederzeit fest zu mir standen und mir den nötigen Rückhalt und die Unterstützung gaben, meine Ziele zu erreichen.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The traditional lecture is one of the oldest approaches to teaching in higher education, practised by many European universities beginning in the 12th and 13th century. The original purpose was to give students the possibility to create their own books by listening to a lecturer and writing down everything he said. After the introduction of the printing press in 1450, the role of the lecture gradually changed. Today, a lecture is less focused on the pure presentation of knowledge. Instead, the lecturer presents a description of key ideas of a subject, explaining various important topics and giving interpretations that often include current research on the issue.

The interiors of the lecture hall also have changed drastically over the last few decades. Blackboards and chalk lost more and more ground to overhead projectors which eventually were replaced by data projectors and computer-driven presentations or electronic whiteboards. Most lecture halls today are equipped with at least one big projector screen, computers, sound systems, cameras and other multimedia features, like DVD players and book scanners. This way, lecturers are able to present knowledge in ways that had not been possible only half a century ago: using a camera and a data projector, for example, chemical experiments can be magnified for all students to see. Movies can be shown to support the understanding of biological or medical issues, and animations can help to explain complex algorithms in computer science.

Despite these improvements, however, lectures are still heavily criticised in literature [Bli00, Gib81]. The main argument against the traditional teaching scenario is the lecture's rigid layout: lecturers present new information to the learners without

guiding their learning processes. Students, on the other hand, are not able to interactively affect the procedure of the lecture without disturbing the lecturer or fellow students massively. The limited interactive possibilities in lectures, however, induce a set of problems regarding students' attention and motivation as well as the adaptivity of the lecturer's instruction.

Some lecturers attempt to overcome these problems by asking questions to trigger feedback on how well the students have understood the presented material, as well as to provoke them to actively participate. In lectures with a large audience this is problematic because only a few students are able to interact with the lecturer in this way. The overwhelming majority, however, will not profit from this form of interactivity. Further problems arise if the lecturer wants to get feedback on how the lecture is accepted by the students and what he or she can do to improve it. In lectures with a small audience the teacher typically deduces this information from the students' reactions, e.g., if they are very attentive or looking bored. In lectures with large audiences, this information is usually gathered by passing out feedback questionnaires to the students at the end of a lecture period. Unfortunately this approach is rather imprecise and does not allow the assessment of individual parts of a lecture. Furthermore, it is not possible for the lecturer to quickly react to problems.

From a pedagogic-psychological view, learning (in lectures) has to be reconstructed as an active process (e.g. [Ern95, Hon96, WC91]). Interactivity represents an opportunity for the learner to take part in shaping the information, communication and learning process rather than remaining a passive recipient; thus, an active involvement of the learners has a great impact upon successful learning [Ram92]. In respect to the learning success in lectures, empirical results state that lectures are not generally ineffective, but they are not suitable for a global knowledge transfer [GB96a, Pet79].

Directly connected to the problem of low interactivity in this form of teaching is the lack of adaptivity of the teacher's behaviour: During the lecture the instructor can only adjust a limited amount of contents or topics of his lecture to the needs of the students. On the other hand, adaptivity is an essential tool in the instructional-psychological context to improve the learning process. By adapting explanations or

curricula to the learners' current state of knowledge a greater efficiency of instruction can be achieved. Empirical studies reveal the positive effects of different learner-centred measures upon learning success [Sas92, Cro77, Bli00].

Finally, an essential problem in lectures is that a continuous attention is required from the learner for usually 90 minutes. This premise is not realistic: Typically, the main attention span is not longer than about 20 minutes [Smi01]. Subsequently, a change of activity must take place for the students to maintain their attention (e.g., switching between lecture and discussion phase). Otherwise, the decreasing mental performance after the first 20 minutes result in an inferior knowledge acquisition (e.g. [SSC+63, Blo53]). However, in the classic scenario, activity changes are generally not intended, but if they are, their success depends exclusively on the ability of the lecturer [Ram92].

But despite its obvious didactical shortcomings, the lecture still is an important and common educational scenario because it also has some distinct advantages compared to other teaching methods. Although addressing a larger number of students simultaneously, it is easy for the teacher to adapt the course to different audiences, topics, timetables and available technical devices. Additionally, a flexible integration into the curriculum without thorough planning can be realized, which would not be possible with e.g. text books.

An especially important factor, though, is the economic aspect: Only in lectures an individual lecturer can impart knowledge to a rather large number of students at the same time. Most universities worldwide would not be able to educate the same number of students with seminars and working groups as they are able to handle with lectures. Even elite universities rely heavily on mass lectures for very popular subjects. Because of this, the often discussed disposal of lectures will not be possible, even in the long term.

We conclude that there are evident practical reasons to improve this learning scenario or to create a new (more interactive) scenario as a replacement.

An innovative approach, presented in this dissertation, is to improve interactivity and to realize a bi-directional, synchronous communication in lectures by equipping the students with small electronic devices such as handheld computers [RP02]. These

devices communicate with the computer of the lecturer and thus allow an exchange of information with the lecturer at any time without disturbing the lecture. The type of information exchanged can be arbitrarily complex, ranging from a simple "virtual hand raising" over detailed feedback to quizzes that may even be counted towards the grades of the students. To avoid cost-intensive modifications of lecture halls, the handheld PCs and the server are connected by means of a wireless LAN.

With this technology, we aim to create a new form of multimedia-enhanced teaching: the Interactive Lecture. For this reason, we have designed and implemented a full-featured software package, consisting of a server and clients for the teacher and the students. In close cooperation with the Department of Educational Science II at the University of Mannheim, we have investigated the didactical backgrounds and motivations for interactive lectures and performed a thorough evaluation of the concept. This includes four major and several minor field studies, evaluations and observations. The results of these evaluations suggest that the interactive lecture indeed increases the overall learning success as well as the motivation of the students and a number of other aspects, which will be explained in detail.

1.2 Outline

The remainder of this dissertation is structured in the following way: Chapter 2 explains the pedagogical background of the interactive lecture, and chapter 3 gives an overview over several technological developments that are needed for the realisation, presenting also a selection of projects with similar intentions. In Chapter 4, the software toolkit WIL/MA is introduced, beginning with the architecture of the general system as well as of several relevant parts. Chapter 5 completes the architecture with details of the implementation of both the server and the clients. Extensive results of our evaluations will be presented in Chapter 6. Latest expansions of the WIL/MA software embracing two new technologies: group support in interactive lectures and participatory simulations are introduced in Chapter 7. Last but not least, Chapter 8 will close this dissertation with a conclusion and an outlook on further activities.

2 Psychological Background

In this chapter the psychological background of the interactive lecture is discussed. After a short definition of the term "lecture", an overview of the history of this traditional teaching method and criticism on it is given. Empirical considerations and studies on various aspects of the lecture and comparisons with other instruction methods follow. Derived from the results of these studies, a discussion about the general usefulness of frontal teaching and ways to optimise or improve lectures concludes the chapter.

2.1 The Lecture

A lecture is a special form of a recitation in the context of higher education where a lecturer intends to present a coherent topic and to comment it in order to deliver insight to the auditors of the complex structure of the scientific consideration [Ape99]. To clarify basic problems of the specific discipline that is to be mediated, only words and different optical aids, in a few subjects additionally live scientific experiments or exemplary law cases are used. Contiguous lectures are merged into a course that is usually attended by the same students for the entire semester.

A central attribute of the lecture is its strictly unidirectional communication: *"learning topics are 'transported' from the consciousness of the lecturer to the students. This distinct activity demands attentive and continuous reception of the students. The most important medium in a lecture is speech; the process of communication runs only in one direction. Interaction between students and lecturer are limited to casual enquiries or prove to be a disturbance."* (translated from [RR83]).

Lectures have been used in universities and colleges all over the world for centuries as an established and popular way of knowledge transfer. Until today, they are one of the most dominant instruction methods to be used in higher education. The lecture timetable of the University of Mannheim for the Winter semester 2004/05, for example, has 338 entries for lectures, which are about 21% of all quoted courses.

However, there are some differences between lectures in different countries in terms of formal aspects (i.e., length and number of students) and their purpose.

In continental Europe, particularly in German-speaking areas, the usual lecture takes about 90 minutes and has the intention to communicate scientific proficiency and research practices and to improve the identification of problems, critical thinking and coherent knowledge [Ape99].

Lectures in Anglo-saxon countries are usually not longer than 50 or 55 minutes. Bligh, who defines lectures as "more or less continuous periods of presentation by a speaker who wants the audience to learn something", identifies the four aims of a lecture summarily as the acquisition of information, the promotion of thought, changes in attitudes and the development of behavioural skills [Bli00]. Not all of these aims are perfectly met, as can be seen later in chapter 2.2.1 on page 10.

2.1.1 A Brief History of the Lecture

The lecture as an academic method of instruction has a long tradition [Ape99, McL76]. Originating in antique philosophy and rhetorics, the roots of the lecture can be traced back as far as the 5^{th} century BC. The common recitation in Greek philosophy schools of this time (*"praelectio"*) is considered a basic form of academic instruction, similar to lectures [Pau66]: in public parks of Athens, Plato gathered his students, read out a canonical text and interpreted it with his own comments.

In medieval times, the lectures in early universities had other intentions, though. Offering a very easy way to transport the knowledge contained in the rare and expensive manuscripts to a large number of students simultaneously, lectures were the most preferred instruction method in these times. However, the lecture was void of any purpose beside the mere multiplication of information. This dictation style became

more and more obsolete after Gutenberg invented the printing press in 1450, and scientific texts became more common.

With the prevalence of the *"libertas philosophandi"* (freedom of doctrine) that also led to the first non-latin disposition in Halle, Germany, in the late 17^{th} century, the role of the lecture changed. The teachers were now requested to present scientific disciplines and impart research outcome in lectures and discussions. Communication of canonical knowledge was only secondary; lectures were primarily supposed to serve the enlightenment of students for independent thinking and to further free scientific research. However, lectures in reality were still little more than a commentary of an underlying script: the lecturer read out a text and interpreted it for the students [Ape99].

Modern universities, having a novel vision of the lecture as basic academic instruction, have emerged at the beginning of the 19^{th} century. The lecturers are supposed to recite freely and to involve the students rhetorically. In Germany, the "Humboldsche Bildungsreform" (around 1810) entailed a rearrangement of universities in Prussia and other German states that is still effective today.

2.1.2 Criticising Lectures

The famous British lexicographer Samuel Johnson has claimed in 1781 that *"Lectures were once useful; but now, when all can read, and books are so numerous, lectures are unnecessary. If your attention fails, and you miss a part of a lecture, it is lost; you cannot go back as you do upon a book."*. As a matter of fact, lectures have been criticised as inefficient instruction method frequently over the last 200 years, primarily though in German and other continental European universities. The discussion concerning the problems of lectures was extended to Anglo-american universities not before the late sixties of the 20^{th} century.

German Views of the Lecture

There are three eras of prominent discussion about the lecture in German universities [Ape99].

In the 19^{th} century, when the concept of the new German university was designed and realised, many critics disliked the lecture because it was absurd to read out monolithic texts which every auditor is able to get a copy of easily. It was also suggested that this leads to the failure of students to learn to deal with a subject autonomously. Additionally many critics asserted that the monologue of the lecturer induces boredom and thus impedes efficient reception of the topic. In 1844, a decree of the Prussian ministry actually demanded that the dialog between teacher and students has to be emphasised in universities.

Around 1900, advantages and disadvantages of the lecture were again discussed vehemently (see chapter 2.3 on page 16), particularly the passivity of the learner and the atrophied auto-didactical skills of the students, which were reduced to note-taking. Many proposals to improve this situation implied a more dialogic character of the lecture and a stronger involvement of the students. Schleiermacher even preferred a lecture according to the ancient Greek archetype, where the teacher is obliged to arouse the students' attention and interest.

Although some critics even demanded to abolish the lecture completely, there were also supporters of this instruction model. The lecture was lauded to give a spirited overview of a topic by a living person, to reference this collected knowledge to real life situations and to arouse interest leading to an independent understanding of the subject.

Beginning in 1960, German universities became educational establishments for the masses. While in earlier eras, lectures were most notably criticised by politicians and professors, now mainly the students demanded changes of the academic instruction methods. It was stated that the lecture does not conform very well to the now overcrowded lecture halls and that it misses recreational breaks, variations in speed and presentation, direly needed visual material, a human touch and - generally - comprehensibility [McL76]. Design and relevance of lecture courses was not adapted very well to the needs of the learners, and the teachers were often accused to have no interest in the learning success of their students. Further criticism included the monotonous instruction style of many teachers, deficient scientific methods, outdated materials, the missing consideration of alternative perspectives and a judgemental position of the lecturers. The lecture was "attacked as symbol of an authoritarian

and therefore undemocratic claim to power" and "downright villainised" [Ape99]. Despite the massive protests from students, however, the lecture and its one-way communication was retained.

Today, academic instruction is evaluated in terms of efficiency, and observed from the perspective of a teaching-learning situation [Dub00]. The problem of the students' divided attentiveness, who have to listen to the docent's monologue attentively on the one hand and take notes on the other hand, was picked up again [RR83]. Also, a discrepancy between the ex-cathedra teaching that is practised in lectures and the anticipation of autonomous learning for life was repeatedly pointed out [Ape99]. The reception of recited lore leads to passive knowledge that cannot be applied to concrete situations; an active acquisition of new knowledge is not supported in lectures.

Anglo-American Lecture Experiences

The Anglo-american approach is much more pragmatic. Lectures are seen purely as an academic instruction method that aims to mediate information about a scientific discipline and to encourage discerning cerebration [Ent81]. Similar to Europe, critics argue that these requirements are not met by the teacher-centred instruction of the lecture; the "authoritarian social situation" [Bli00] hinders autonomous thinking. Primarily, however, the efficiency of lectures regarding advantages and disadvantages and the achieved learning success is being revised.

Anglo-american research shows that lectures combine rhetorical and didactic elements. Lectures are seen as an essential but problematic instruction style and depend heavily on the didactic skill of the teacher that can be seen in his or her ability to consider the purpose, content and method of a lecture as well as situational conditions and the requirements of the students.

Generally speaking, the purpose of the lecture is to merge a logically precise interpretation of a subject with a stimulating presentation.

2.2 Empirical Considerations

In Germany, there are many different propositions about the effects of lectures, and also theoretically founded claims, but these statements are usually based on personal experiences. Empirical results, however, can primarily be found in Anglo-american research. The focus of most studies is the comparison of lectures with discussion-based instruction styles. The studies particularly deal with performance measurements (e.g., exams), but also motivational aspects like attention and acceptance are regarded.

A recent survey of German lecture research can be found in [Ape99]; for Anglo-american research, see [Bli00]. However, it has to be taken into account that many studies are not published for the lack of significant results [McK68].

2.2.1 Performance of Lectures

A very critical performance index of an instruction method is the *learning success*, i.e., the ability of students to remember the covered topics. When reviewed under laboratory conditions, however, several studies show that verbally presented subjects are forgotten very quickly [Bar32]. Furthermore, the reception of the information in a lecture declines over the time: when using a tape recording of a lecture, the knowledge assimilation of most students was considerably alleviated after 15 minutes and almost zero after about 30 minutes [Tre51].

The dire results regarding the learning success of the cardinal presentation method in lectures can also be validated in real life experiments. In 1923, a large scale study with almost 750 psychology students was carried out with the result that students remember about 60% of a lecture when tested immediately afterwards. However, eight weeks later, only an average of about 20% of the knowledge was still present [Jon23].

In another study in 1976, the learning success was as low as 40%, measured directly after a lecture. It did not make any difference if the students were told that the topics were part of the exam (Group 2, “motivated lecture group”) or were not given any additional information (Group 3, “unmotivated lecture group”). Group 1 was the

comparison group ("motivated reading group") where the students had to learn the subject they were told to be part of an exam by themselves from a book. The students in this group were able to remember about 50% of the subject afterwards. Four weeks later, however, all students could recall an average of 35% of the material, regardless which group they were in [McL76]. McLeish explained this "equalisation effect": *"Irrespective of differences due to the teaching method used, the work which students do for themselves in preparation for an examination will tend to bring their scores close to equality. This will vitiate any long-term comparison of the relative efficiency of the various 'treatments' to which they have been subjected" (from [McL76, pp. 271]).*

Indeed, in many comparative studies, the lecture did not perform much inferior to other instruction methods like discussion groups or seminars in terms of mediation of factual knowledge, although more interactivity was usually preferred by the students. Joyce and Weatherall showed in a controlled experiment that discussions have a slightly higher success, but the difference was so small that the lecture was rated more efficient because the economic aspects overweighed. For a short overview over the conclusions of a sample of representative studies regarding the reception of information, see Table 2.1 on the following page.

When it comes to the encouragement of critical and autonomous thinking however lectures are usually quite inefficient. Although "promotion of thought" is defined very different in literature ("flexibility and creativity", "multiperspective perception", "quantity and diversity of ideas" or "depth of questions" to name but a few), lectures tend to be rated inferior to almost any other instruction design. The results of the studies for this criterion are summarised in table 2.2 on page 13.

This inferiority can be explained by the fact that students can solve problems not by the application of principles alone; they have to combine these in order to create principles of a higher level. To do this, the students must exercise and actively apply the received information, which is not intended in traditional lectures [Gag65].

Another performance issue is the ability of an instruction method to change the students' attitudes. Regardless whether the intention is to mediate intrinsic values and virtues, to effectuate changes in personality or social adaptation or to arouse

Instruction method	Lectures less efficient	No significant difference	Lectures more efficient
Personalized learning (similar to eLearning)	20	17	8
Discussion (different types)	18	54	22
Textbooks, autonomous learning	10	21	9
Enquiry (e.g., projects)	6	6	3
Others (e.g., multimedial instruction)	27	57	20

Notes:
The values given in the cells are the total number of accounted studies with the denoted result; i.e., of all studies comparing the lecture to a discussion, 18 rated the lecture less effective and 22 as more effective.

Table 2.1: Summary of studies comparing lectures with other instruction methods in terms of learning success

interest, the lecture is generally inapplicable. Particularly, the lecture is not suited to win the students for a subject; to motivate the audience therefore should not be the main objective of a lecture [Bli00].

Developing behavioural skills, the fourth aim of a lecture as defined by D.A. Bligh, can at least be supported by a lecture. To learn the "knowing how", the students must first be communicated the "knowing to", i.e. factual knowledge. While the lecture may be inept to mediate the "knowing how" properly because it is not intended to practise a behaviour in this context, it is much more efficient to deliver the fundamental information.

2.2.2 Learner Prerequisites

Studies regarding the performance of lectures usually fail to take the varying requirements of the students and the differences in their personalities into account. This, however, is a very important aspect to be considered when comparing instruction

Instruction method	Lectures less efficient	No significant difference	Lectures more efficient
Discussion	29	1	2
Textbooks, autonomous learning	1	3	1
Enquiry	5	1	1
Others	12	17	0

Notes:
The values given in the cells are the total number of accounted studies with the denoted result; i.e. of all studies comparing the lecture to a discussion, 29 rated the lecture less effective and 2 as more effective.

Table 2.2: Summary of studies comparing lectures with other instruction methods regarding the encouragement of autonomous thinking

methods. Simple comparisons are too global, instead the learning process in a face-to-face situation between teacher and individual students has to be examined more carefully [McL76, LM96].

A comparison between group instruction and lectures, for example, showed that group instruction entailed a better recognition and comprehension for good students; weak students, however, had better results in the lecture [War56]. Less competent students prefer to learn with stern guidance, whereas efficiency-oriented students favour autonomous learning instead.

Results like these show that lectures may be a very reasonable form of instruction for part of the students, while for other students they are rather inefficient. This means that not only the instruction method and the educational aim have to be matched, but also the personality of the student.

2.2.3 Attention and Concentration in Lectures

To get a better insight into the deficits of the lecture regarding concentration and attentiveness, Bloom examined in 1953 the thinking processes of learners during a

lecture in comparison with those during a discussion [Blo53]. He used the "stimulated recall" method where a recoding of the prior disposition was played which was paused at certain places to ask the the students what they thought at that precise moment. The results showed that the discussion was superior to the lecture in terms of creative and reflective thinking: 31% of the students' thoughts during a lecture were irrelevant with respect to the topic, but only 14% during the discussion.

In 1970, Schoen used the same method to compare seven instruction methods [Sch70]. He discovered that during a lecture only two thirds of the students were attentive, though the other methods fared not much better. The highest attention was measured during movie screenings and problem solving exercises, the attention was very low during panel discussions and tutorials.

The attentiveness, however, is directly correlated to the reception and assimilation of information [SSC+63]. Therefore, studies were required to measure the course of the attention over the time span of a lecture.

Lloyd proved in a number of experiments the assumption that the reception of knowledge in lectures is hampered by the decline of attentiveness [Llo68]. He declared the receptivity of students as a variable over time and wanted to show that the concentration or performance of the students is directly interrelated. The variation in receptivity was measured by observation, personal information of the students and inquiry of the teacher. To record the decrease of concentration, he interviewed the lecturers. The quality of the assimilated knowledge was constructed from a comparison of the teacher's script and the notes of the students. With this data, Lloyd constructed the chronological course of three variables: "transmittal performance of lecturer", "receptivity of class" and "assimilation by class" that is shown in Figure 2.1 on the facing page. He discovered that the sequence of performance during a working period [McL76] can be applied to the Anglo-american 50-minute lecture, too: after an "initial spurt" with a very high performance, the students encounter a "middle sag" resulting from a mixture of boredom and weariness which is followed by an "end-spurt" with a level of performance almost equal to the beginning. The end-spurt can be explained with the activation induced by the near end of the lecture; the sudden drop-off shortly after is due to discursive thoughts about activities after the lecture (e.g. next lecture, visit of the cafeteria).

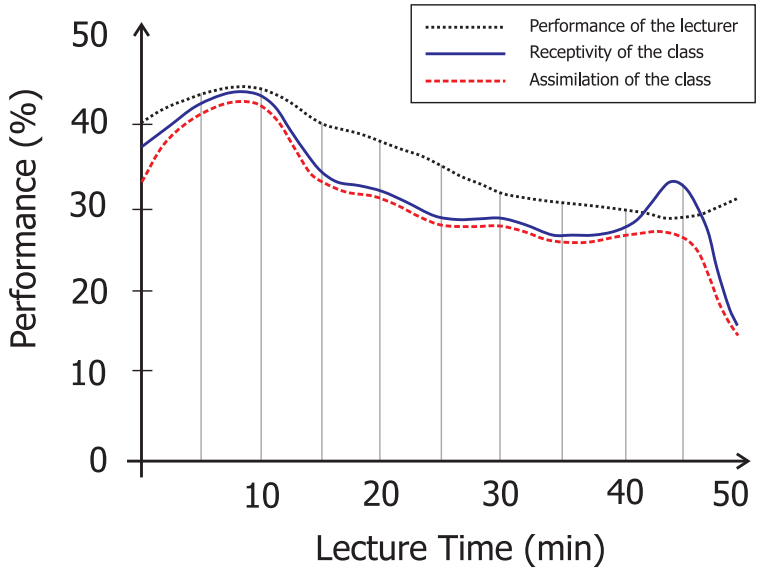

Figure 2.1: Course of the attention of students and teacher during a 50 minute lecture (from [Llo68])

With the measurement of the heart rate of the students, Bligh confirmed the results of Lloyd. 14 students listened to a 70 minutes lecture, observed a simulation passively for 20 minutes and then participated in a discussion for another 60 minutes. During the lecture and the discussion, the heart rate of four of these students was measured every 5 seconds. The graphs of his study can be seen in Figure 2.2 on page 17. He assumed that there are two distinct periods in a lecture: the first 20-30 minutes, where attentiveness starts high and drops rapidly, and the rest of the lecture, where attentiveness keeps steadily decreasing at a low level.

It is assumed that this typical course of the lecture can be altered by didactical means of the teacher: *"The initial fall in class receptivity (period 10 to 20 mins.) is frequently due, it would seem, to mental confusion. This can be relieved by a brief period of recapitulation, consolidation and example to clear the student's mind possibly with the use of audio-visual aids" (from [Llo68, pp. 25]).*

2.2.4 The Students' Opinion about Lectures

Several surveys in both German and Anglo-american universities show that the lecture is quite unpopular with most students. There are certain individual differences,

though; older students, for example, are less negatively attuned to the traditional instruction method than younger students [McL70].

Still, the lecture is accepted as an inevitable part of the university; even in a survey in 1969 in Germany during a time of very critical discussions about the lecture, the interviewed students usually stated that they considered the lecture to be necessary. They indicated, though, that they demand a concise and tight format of the subject and a didactically advised presentation [Hae69].

2.3 Disadvantages and Advantages of Lectures

In the following section, a survey of disadvantages and advantages of lectures is given, which can be derived from the various previous considerations.

2.3.1 Disadvantages of the Lecture

No Active Involvement of the Students

Students in the lecture as a teacher centred instruction method are usually "sentenced to passivity" [RR83]. This is one of the most criticised aspects of the lecture: it is very hard for the lecturer to activate the students in large lecture halls where the students are barely able to ask questions or to give other feedback. Furthermore, the inhibition to pipe up in front of the fellow students is very high - especially in large lectures - and it is also very difficult for the lecturer to answer properly without losing too much time [Car01].

The small number of interactive possibilities in lectures accounts for a low adaptivity of the lecturer's talk as well as for a limited attentiveness and motivation on the part of the students.

No Adaptivity of Instruction

In a lecture, the teacher has to do without a valid micro-feedback regarding the comprehension of the students or their acceptance. He or she is forced to receive

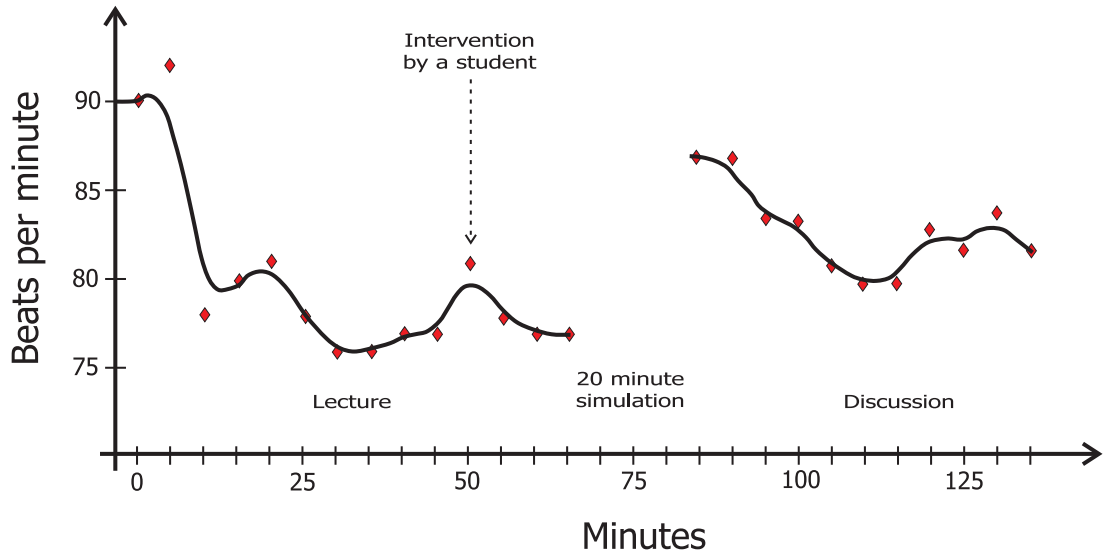

Figure 2.2: Attention of students in a class measured with heart beats per minute (from [Bli00])

this information from other sources which are either not timely (e.g., with surveys *after* the class), biased (e.g., by weighing the criticism of a few complaining students too high while the rest of the students are very content) or inaccurate (e.g., by only observing the students sitting in the front rows).

Based on this information, an overstated adaptation of the lecture's speed or niveau would be precarious. Furthermore, in very big classes it is almost impossible to match the requirements of all the individual students' personalities adequately.

Lack of Attentiveness

In Germany, a traditional lecture demands 90 minutes of high attentiveness from the students and the teacher. As can be seen in chapter 2.2.3 on page 14, however, the mindfulness and thus the reception of the students begins to drop already after about 20 minutes. Even to expect the students to be attentive for only 50 minutes (Anglo-american lecture) is supposed to be "absurd" [Bli00].

Attentiveness can be achieved by entertaining or tantalising components; this, however, depends heavily on the skill of the teacher. Another possibility to improve attentiveness is to involve the students stronger in the lecture with innovative methods [MK75].

Lack of Motivation

There are also motivational problems in lectures [Rhe00]. To be able to get performance feedback is a necessary requirement for performance-oriented behaviour [HSS85]. The anticipated pride of the own success is an important incentive for motivation which is a much more important factor for performance than intelligence or social background of the students. In traditional lectures, however, feedback about the students' learning success is scarce.

Studies show that an active involvement of learners has a positive effect on motivation. A high motivation yields a longer and more thorough preoccupation with an exercise: cognitive processes, important for the construction and conceptualisation of knowledge, are facilitated.

Lack of Long-term Knowledge Assimilation

The students remember the subjects of a lecture only for a very short time; for a longer recollection, other instruction methods are more appropriate [GB96a]. Additionally, lectures are not very suited for other learning aims besides the transfer of factual knowledge, such as change of attitude or improving analytic and judgemental skills.

Learning, on the other hand, is a process that begins with the construction of knowledge. This knowledge then has to be used in situational exercises which encourages the automation of further learning processes [Dub95]. Using the terminology of Weinert [Wei98, Wei99], learning first occurs in vertical, then in horizontal and finally in lateral direction. Vertical transfer means systematic development of fundamental knowledge; a process direct instruction methods like the lecture are most suited for.

2.3.2 Advantages of the Lecture

Economy and Flexibility

An important advantage of the lecture is its ability to transfer knowledge to a large number of students simultaneously. In terms of "mass universities", lectures are therefore necessary to mediate basic knowledge which can be refined later by other parallel instruction methods. McLeish says about the lecture: *"As a teaching device,*

it [the lecture] is undoubtedly the most economical method by which an individual can present in a personalised and continuous argument the general framework of understanding for fundamentals of a particular subject, emphasising the key concepts and involving the audience in reflective thought that moves in time with the on-going performance. An air of studied improvisation gives the lecture its salient character, that is, an extended conversation that has developed into a monologue contribution to the master" (from [McL76, p. 253].

The only instruction method competing with the lecture with regard to efficiency is autonomous learning with a textbook, but this is much less flexible in terms of adaptivity to different audiences, updated topics or variations in multimedia equipment. Even compared to other common instruction methods, which have to be much more carefully planned and adapted to certain environmental parameters and audiences, the flexibility of the lecture has to be emphasised [GB96b].

Presentation of Current Research

The high flexibility of lectures also allows using them for the presentation of current research results. Not yet published topics can be introduced which on the other hand may have positive effects on the motivation of the students. Furthermore, the lecture is the best method if the subject is not available otherwise in an acceptable version, e.g., as textbook, or if it has to be adapted to a certain group of students. In computer science, for example, a close correlation between the subjects of a lecture and the exam is quite common because text books in this discipline are usually outdated in a very short time.

Concise and Accentuated Presentation

Lectures are very useful to arouse interest or motivate students by giving an overview of a discipline which is then dealt with in a number of follow-up courses. With the lecture, important aspects can be accentuated to give the students a comparative presentation and evaluation of different scientific opinions.

Additionally, it is often stated that students - particularly in lower semesters - are in many cases too immature to learn efficiently with autonomous methods [McL76]; the lecture is needed in these cases to transfer the necessary knowledge. This is among

other things due to the fact that the lecture effectively breaks down the subject into a number of smaller portions which are presented at regular intervals. This way, the continuity of learning is improved opposed to learning with a static text [USG96].

Enthusiasm and Motivation

If a teacher is very enthusiastic about a certain topic, this enthusiasm can vest in the students as well. A lively lecture is an "aesthetic delight" and may arouse interest and active cogitation about the subject [BJP76, MLMI66].

Interpersonal Contacts

Studying at a university not only involves the acquisition of knowledge, but also the establishment and maintenance of social contacts. Lectures are very useful in that context: the lecturer is able to be present and to learn more about the students. On the other hand, the students have the chance to meet the teacher in person, whereas the authors of textbooks are intangible schemes for most of the readers. Even other more personal forms of education - like seminars or discussion groups - usually do not offer the chance to meet *all students of a class at once, but only a small group.*

For the same reasons, the students of a class are able to get to know each other on a much larger scale to form learning groups or to spend the time together for other reasons.

2.4 Optimising and Improving Lectures

The preceding discussion has shown that the lecture has many advantages and thus deserves to be preserved. Still, there are a number of disadvantages to be considered; the most important is the lack of activation and interaction of students which leads to most of the other described problems. By improving the interactivity the efficiency of the lecture should rise also by enhancing motivation, attentiveness and in the end the assimilation of knowledge for the students. In order to do this, it is necessary to have detailed information and feedback available.

Different approaches in instructional theories feature listings of necessary elements for a "direct instruction" and how to use them. Methods that create the chance to actively process information, to continuously survey the understanding during the lecture and to get timely feedback are regarded as very important [Rei99, Mer91, Gag65].

Feedback of students enables the teacher to adapt the lecture to the needs of the audience and thus to improve the learning process. Prepared topics can be altered, completely changed or replaced in later lectures. Furthermore, the style of presentation can be adapted. While it is not possible to match the lecture to the individual personality of each student, a more learner-centred instruction style can be achieved. During school instructions, for example, it was observed that an immediate feedback during the lesson is clearly superior to a delayed feedback in form of the discussion of the homework [KK88, van80].

Learning is an active process that depends on the readiness of the students. Thus it is obvious that an active involvement of the learners has a high impact on successful learning [Ern95, Jon94, Hon96]. In the following, different possibilities to optimise traditional lectures with feedback for teachers and students are presented.

2.4.1 Enabling Feedback during a Lecture

Quizzes and Tests

Since adaptivity requires that the teacher is informed about the state of knowledge of the students, it seems reasonable to include small knowledge tests or quizzes during a lecture. It is also considered an essential way to maintain or even raise attention by providing new stimulations in regular intervals by the integration of the possibility to pose questions or to discuss a topic [Bli00].

Exercises during the first 30 minutes after the beginning of a presentation are relevant for the consolidation of knowledge. But the occupation with questions covering the current topics during a lecture has other advantages as well [Ber68]:

1. Individual aspects of the current subject can be emphasised.
2. Trying to find the answer to a question is an application of the new knowledge.
3. The learner can check if he or she has understood the subject correctly. If not, an immediate discussion of the answer can have a positive effect on the motivation because the student may have a better comprehension of the following topics.
4. The attentiveness is improved by changing the media.
5. A quiz may be considered a small break, in which the students have some time to recapitulate the new information.
6. With excercises, older knowledge can be refreshed.
7. The teacher gets feedback on the current level of understanding in the class.

Short quizzes with a number of multiple-choice questions five minutes to the end of a lecture can improve the memorisation of information immensely. This was proven in a study in 1923 where students of one course were allocated to five different groups: Group A was tested immediately after a lecture for their memory performance, group B on the next day, group C after a week, group D after two weeks and group E after three weeks. The results shown in Figure 2.3 suggest that the quiz seems to provoke a consolidation of new knowledge, and the sooner this happens, the better the students can remember the subjects.

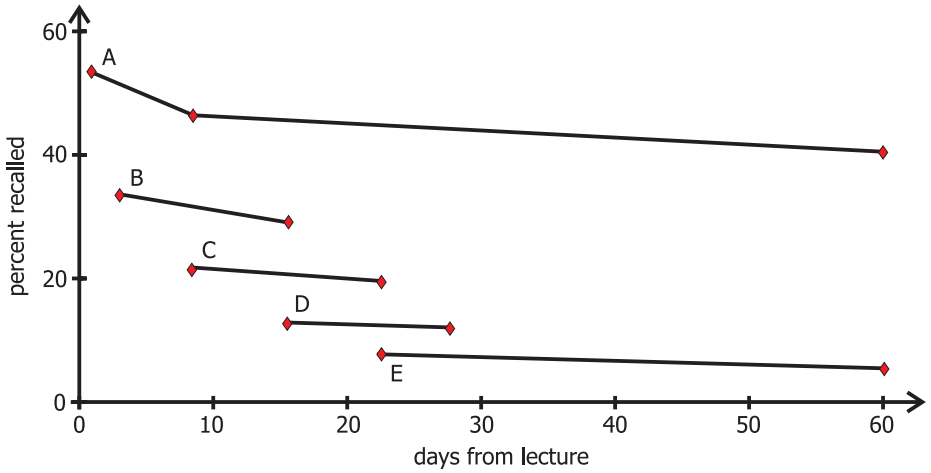

Figure 2.3: Results of Jones' experiment regarding memory performance (from [Bli00])

Different methods to apply multiple-choice quizzes most efficiently were tested. Several professors in the UK and the former USSR used panels with different buttons, each representing a possible answer to the question. The students picked their answer, pushed the corresponding button and by doing this turned on a lamp on a public panel. By the distribution of lights, the teacher and the students could see how many participants had chosen the correct answer and thus if the current topics was understood well. Other methods included cubes with differently coloured faces or coloured cards. Although the preparation of the quizzes and their realisation during class takes a lot of time, the deeper understanding of the subject outweighed the higher cost.

Measuring the Acceptance by Students

Feedback not only embraces just the learning success of the students but also other aspects, like the current understanding of the subject, the current attentiveness or the ability to concentrate. This feedback can help the teacher to better adapt the lecture to the requirements of the students. If the lecturer observes that the students' present attentiveness is very low, he or she can insert multimedial elements or reschedule an important subject to the next lecture and talk about less relevant topics instead.

Students' Questions

Spontaneous questions of students are also a reasonable way to get feedback. Answering these questions leads to a variation of activities just like a quiz, with similar positive effects on motivation and attentiveness. Furthermore, the ability to ask questions is sufficient to enhance the concentration of the students because they tend to be more engaged in the subject than without this possibility.

2.4.2 Introducing Computer Supported Feedback in Lectures

Gathering feedback in traditional ways, i.e., with hand-raising, speech or questionnaires, has two distinct problems. First, the feedback may be tainted, because it is impossible to let the students answer individually during class without any form of tools. When asking a question like "Raise your hand if you think this solution is

correct!", as it is often done as motivational element during a lecture, unsure students will tend to answer in accordance with students they think to know the answer. Thus the lecturer gets the opinion that everything has been understood well although this is only true for a minority of students.

The second problem is that there is no way to let the students interact anonymously. Most students will not participate openly in a large lecture hall at all when they are given the chance, because they fear to disgrace themselves in front of their fellow students by giving a wrong answer or asking a stupid question. Thus, even if the teacher tries to motivate the students by asking questions or giving the students the chance to say something, he or she will only reach a small fraction of the class while the rest of the students remains inactive as in the remaining part of the lecture.

The only way to solve these problems with traditional utilities is to hand out forms to the students to fill in their answers, questions or other kinds of feedback. In this case, however, the information (the input from the students as well as the response by the teacher) is outdated for both, the lecturer and the students, because it will only be considered when the actual lecture is completed and all participants are already preparing for the next topic.

For these reasons, prior studies using quizzes already fall back on technical aids like button panels or coloured cards to help the lecturer aggregate and estimate the feedback more accurately. But to receive the feedback and to respond accordingly requires much attention and commitment from the lecturer: The more students participate in a course, the sooner the lecturer won't be able to handle all the information presented to him or her. Furthermore, these simple tools only allow for a simple kind of quiz, and thus have only a very limited usability. For other forms of feedback it may even be impossible to find such simple solutions at all, e.g., to allow students to ask questions anytime anonymously and without disturbing the lecture would require a complicated messaging system.

A viable solution to enable interactive and synchronous feedback during class is the introduction of modern computer technology into the lecture. Small input devices handed out to the students are used to collect the feedback (answers to quizzes, acceptance, questions). The gathered information is forwarded to a central instance

that aggregates and analyses the data for the teacher. The lecturer is presented a representative and continuously updated summary of the collected data, so that the actual number of students is irrelevant.

To build and evaluate a system that can be used to create a lecture with electronically enhanced interaction potential - in short an interactive lecture - is the content of this dissertation.

3 Technologies and Related Work

This chapter begins with a short description of three technological advancements needed to realise the concept of interactive lectures. In the second half, projects with similar interests or ideas are introduced and discussed.

3.1 Technical Prerequisites

Two major technical developments were necessary to be able to create an interactive lecture as introduced in this dissertation. First of all, small, powerful computers were needed that can be carried around easily and do not hinder the student during the lecture, as desktop PCs and even Notebook PCs most probably do. Also, a reliable wireless communication was required to connect these devices with each other and/or with a central server, regardless of the student's position in the lecture hall.

3.1.1 Pocket Sized Computers

After the invention of the first laptop by Adam Osbourne in 1981, it took more than a decade before the first pocket sized computer was released; until then, the only programmable devices that would fit into a pocket were scientific calculators from Texas Instruments and Hewlett Packard. The early pocket computers were named "Palm Zoomer" or "Psion" and they all were proprietary in hardware and in software.

In 1996, a new generation of pocket computers was introduced. Almost all of these devices can be divided into two groups that are both programmable and remarkably

powerful, even compared to desktop PCs. The computers of the first group use PalmOS as the operating system and were initially published by only a single company (U.S. Robotics). The other group was more heterogeneous with regard to the publisher: Casio, Hewlett Packard and other companies produced pocket computers for which Windows CE was generally used as the operating system.

The remaining devices which neither used PalmOS nor Windows CE disappeared from the market after only a few years. The most notable representative of this faction was the Apple Newton that was renowned for its excellent handwriting recognition, connectivity and expendability. But being larger, heavier and more expensive than the competition, its product line was discontinued in 1996 after the release of only three models, the "MessagePad 130" being the last.

PalmOS

The first devices featuring the PalmOS operating system were the Pilot 1000 and 5000, both released in 1996 and quickly replaced by the more renowned Palm Pilot in 1997. Beginning with the Palm Pilot, pocket sized computers were named Personal Digital Assistants - PDAs. The Pilot was the first device in the typical layout of a PDA: A small, square screen in the upper half, a smaller input area below and four to five buttons at the bottom make up the front of the device that featured no keyboard at all. Instead of that a pen ("stylus") is used to enter text (using simple script recognition) and to point somewhere on the screen.

Two years later, the operating system was licensed and other companies, most important Handspring and Sony, started to develop PalmOS compatible PDAs. PalmOS - today in its 6^{th} version - always carefully matched and supported the specific properties of the small devices, providing a very intuitive user interface, best utilisation of the small screen space, maximal battery lifetime and a very efficient memory allocation [BP02]. On the other hand, many paradigms known from the PC world did not exist, e.g. a file system, multitasking or flexible windows. Thus, it was very hard for programmers to develop software for PalmOS computers and almost impossible to port existing software from other platforms to PalmOS. Some of these shortcomings were eased a bit in the last two major releases of the operating system, but still the

Figure 3.1: PalmOS based PDAs: the original Palm Pilot 1000 (left), Sony Clié, the first licensed PalmOS compatible PDA (middle) and a modern Palm computer, the Tungsten T2 (right)

basic concept was maintained. This, however, leads to a huge disadvantage: there are still no fully compatible runtime engines for major platform-independent computer languages available, in particular for Java.

PalmOS compatible PDAs used the Dragonball processor for a very long time before most vendors switched to the much faster Intel PXA, which is the core of almost every PDA at the moment. Their static and monolithic operating system made it very hard for hardware providers to create extensions, so until now only very few PDAs have Bluetooth support and even fewer use Wireless LAN for connectivity. A selection of PalmOS compatible PDAs is shown in Figure 3.1.

Windows CE, PocketPC, Windows Mobile Edition

The second group of PDAs started out as so-called handheld computers. Contrary to the typical design of the Palm Pilots, handheld computers still featured a keyboard and the screen was twice as wide as it was high. To access both screen and keyboard, the device had to be opened. Because of the cumbersome handling, the almost unusable tiny keyboard and the unconvincing size, handheld computers quickly became obsolete when as early as 1998 the first PocketPC was released. It had the same layout as the Palm Pilot and was compatible to prior Windows CE devices.

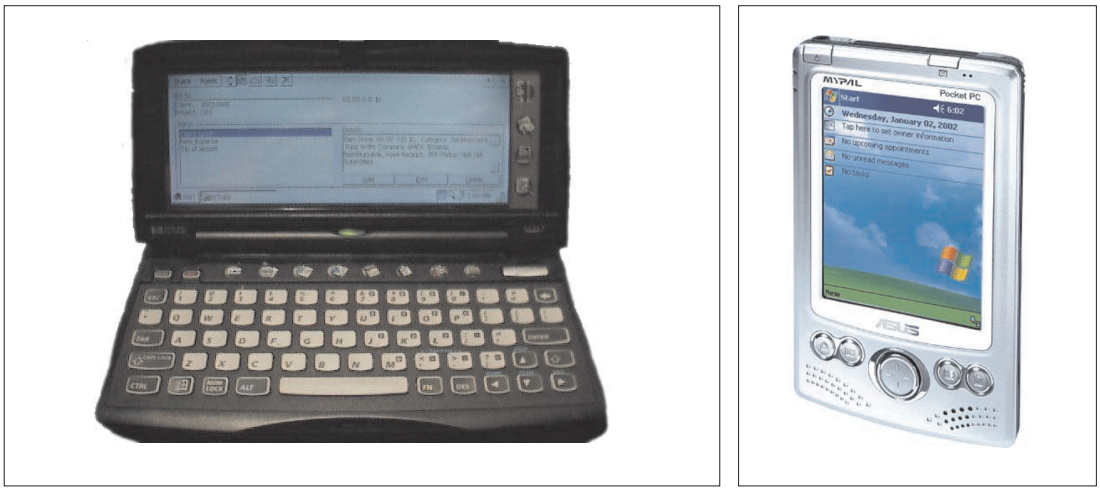

Figure 3.2: Windows CE devices: a 1996 handheld computer from Hewlett Packard (left) and a modern PocketPC, the ASUS MyPal 620 (right)

Many companies - among others Casio, Hewlett Packard and Hitachi - produced PDAs running Windows CE, the first multitasking operating systems with windows that was available for PDAs. Beginning with Windows CE v4.0, the operating system was renamed to "PocketPC". After only two versions (PocketPC 2002 and Pocket PC 2003), it was renamed again to Windows Mobile Edition. Examples for handheld computers and PocketPCs are shown in Figure 3.2.

Windows CE was a true multitasking system, featuring windows and other well-known paradigms from graphical user interfaces on standard PCs, a modular kernel with exchangeable drivers for peripherals and a high adaptivity to both hardware and key parameters like screen size, colour depth and input method [Mue00]. This way, it was very easy to create software and hardware for these easily expandable PDAs. Unfortunately, the metaphors of desktop PC user interfaces are not optimal for PDAs. Although all features provided by PalmOS PDAs were also offered by the PocketPCs, the user interface was obviously inferior. Furthermore, other problems like low battery life time, memory consuming software and quickly overloaded processors are evident.

The advantages of the operating system, in particular the easy development of software, were initially derogated by a multitude of processor types that found their way into the different devices. Fortunately, one of these CPUs, the StrongARM

processor produced by Intel, proved superior to the others. Later, the StrongARM was replaced by the compatible Intel PXA, so both PalmOS and PocketPC PDAs now use the same core.

Due to the open architecture of a PocketPC [WH01], many devices are available with built-in Wireless LAN or Bluetooth support, most others can be expanded with inexpensive Compact Flash or PCMCIA adapters. This fact and the availability of Runtime engines for Java and other platform-independent languages make the PocketPC an ideal mobile device for interactive lectures.

The typical PDA today[1] - regardless of the operating system that is used - has about 64 MB of RAM (extendable up to 4 GB), a high resolution colour display (up to 480x640) and about the computational power of a standard PC in 2001.

3.1.2 Wireless Communication

With mobile computers, i.e., notebooks, PDAs, subnotebooks, etc., becoming smaller and smaller on one hand and connectivity becoming more and more important on the other hand, the desire for wireless communication has grown considerably. Today, a large number of different systems are available: Bluetooth, Wireless LAN, InfraRed communication, GSM and UMTS to name but a few.

Communication with infrared senders and receivers (IrDA) was one of the first successful options to transfer data between two devices just by placing them near to each other. PDAs can be synchronised with PCs, print jobs can be transferred directly from the notebook to the printer, and managers can exchange their virtual business cards with their PDAs. A huge disadvantage with IR, however, is that only direct peer-to-peer connections are possible, and an unobstructed line of sight between sender and receiver is required [KB04, VBB03]. Also, the range of IR is very short, so in large classrooms with many students (and therefore many obstacles) IR can not be used for interactive lectures.

[1] as of January 2005

Since GSM and UMTS are licensed techniques [Hil01], provided only by telephone companies and therefore quite expensive, we focused on the two radio network techniques that do not require licensing and thus do not have additional costs: Wireless LAN as specified in the IEEE 802.11 standard bouquet and Bluetooth. Both use the royalty-free frequency band of 2.4 GHz.

Bluetooth

Bluetooth owes its strange name to a famous Scandinavian king, Harald Bluetooth, who lived in the 9^{th} century and united the Danish people under a common Christian empire. What Bluetooth did to his country, a consortium of five leading computer companies wanted to do to different emerging wireless communication techniques. So in 1998, IBM, Ericson, Intel, Nokia and Toshiba started to work together on a unified standard they called Bluetooth. Later, after bringing the *Bluetooth Special Interest Group (SIG)* into being, other companies joined in the development [BS05].

Bluetooth's main purpose is not to be a replacement for Ethernet or other wired networking techniques. The protocol rather aims at connecting peripherals to computers of any kind. In so called *pico networks* a master and up to seven slave devices can share one or more services with each other. A typical scenario for a pico network could be a desktop consisting of a mouse, a keyboard, a printer and a modem, all equipped with Bluetooth modules. As soon as a Bluetooth capable notebook is placed on the desk, it automatically gets connected to these peripherals. Other applications are wireless headsets, car audio systems for mobile phones and synchronisation of PDAs with desktop PCs.

Multiple pico networks can be clustered to a bigger network ("scatter network") with up to 256 devices by connecting a slave to multiple masters or by connecting a master as a slave to another pico network. This way, a construction similar to a common network can be established by implementing proper Bluetooth services offering IP-like interfaces. In Figure 3.3 on the facing page, all three possible network modes of Bluetooth are shown.

Unfortunately, when Bluetooth was published in 2002, it was already outdated. The data rate was limited to 721 kB/s in the asymmetric communication mode and 432

KB/s in the symmetric communication mode (compared to 100 MB/s full duplex Ethernet that was already quite common at this time). Also, the range was as short as 10 meters for Class II devices (mobile phones, input devices) and 100 meters in open space for Class I devices (notebooks, PCs) [MB01, Sie00].

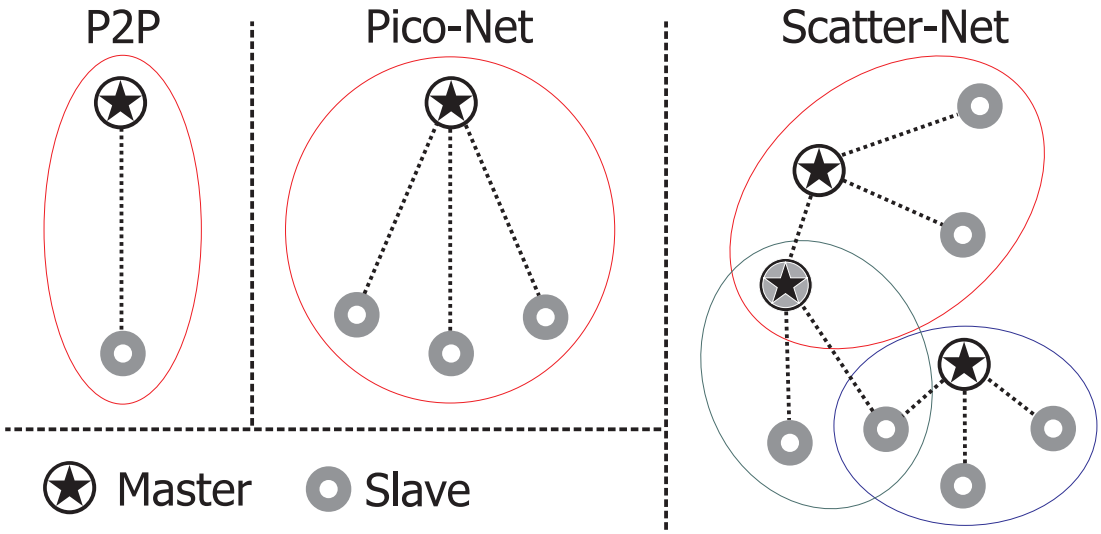

Figure 3.3: Networking with Bluetooth: point-to-point (left), small pico networks (middle) and larger scatter networks (right)

The next version of Bluetooth is under development right now. It promises higher bandwidth, higher range, higher security and better usability.

Wireless LAN

The 802.11 committee started its work in 1990 to develop a standard for a wireless network protocol. In 1997 finally, the first standard for wireless LAN was released: 802.11 [OP99, IEE99b] allows data rates up to 2 MB/s and has a range of up to 200 meters outdoors and 70 meters indoors. Wireless LAN was quickly accepted by hardware manufacturers because it allows effortless switching between wired and wireless networks without having to modify middleware or applications.

Wireless LAN features 11 to 16 channels (depending on the country where the hardware is used). All channels can be used in parallel, but usually every other channel is skipped to reduce radio noise. Without any infrastructure, two 802.11 capable devices may establish an ad-hoc network to communicate peer-to-peer. With

the help of access points, an infrastructural network can be established, which almost any number of devices can connect to at any time. Bridges, finally, link the wireless network to other LANs or the Internet.

Because of its low bandwidth, 802.11 was replaced after less than 18 months by its successor 802.11b [IEE99a], allowing up to 11 MB/s. In 2002, the latest addition to the standard bouquet, 802.11g [IEE03]was released, which improved the bandwidth again. Currently, wireless LAN offers bandwidths up to 54 MB/s, which is more than sufficient for usual Internet activities but still noticeably less than wired Ethernet with 100 MB/s because the protocols used in 802.11 cause an immense overhead.

Quite a problem is the security in wireless networks. 802.11 specifies a protocol extension called *WEP* to encrypt data transmission between two partners, but the encryption used is very weak and easy to break [BGW01]. Furthermore, without additional measures it is easy to get access to a wireless LAN that should be closed to the public.

Since in interactive lectures the wireless networks are usually not connected to the Internet and security is properly handled by the software, wireless LAN is a very stable and reliable method of connecting the mobile devices to the server. The bandwidth of 802.11b is sufficient to handle up to 100 and more students, and a standard access point is adequate for even large sized lecture halls.

3.1.3 Java

A third technical prerequisite concerns the development of the software. A computer language is needed that allows using the developed software on as many different platforms as possible without much effort. The reason for that is revealed by a closer look at the different types of computers that may be used as part of the interactive lecture.

The computer on which the server software runs may be a notebook, provided by the lecturer, or a desktop PC that is installed in the lecturer's desk. On this computer, several different operating systems are imaginable: Windows, Linux, Solaris

and MacOS to name the most popular ones. While it is possible to use another computer that is compatible with a platform-dependent software, this is not as easy with the devices used by the students.

The students are encouraged to bring their own devices to the interactive lecture if possible. Mobile devices, however, may be notebooks with different operating systems, PalmOS compatible PDAs and PocketPCs with as many as four different CPUs, various operating systems, deviating hardware concepts and differing graphical capabilities. Even mobile phones may be a reasonable alternative in the near future.

A viable solution to this dilemma is, of course, to develop the software in one of the major platform-independent languages. The most famous of these is Java.

Java was first released in 1995 by Sun [Sun05]. The basic idea behind the new language was simply that it is supposed to run on all devices. Not only all kinds of PCs, Notebooks, PocketPCs or mobile phones, but also on intelligent tags, vehicles, refrigerators, television sets, toasters and many more.

Java is an object-oriented language that is aligned with C and C++, but easier to handle. From the beginning, Java was shipped with a very extensive library, covering mathematical methods, GUI development, networking, input/output and many other areas. Memory management is handled by a garbage collector that removes obsolete objects automatically. Also, Java does not support the direct disposition of object vectors; instead, much easier traceable (but less powerful) constructs are used. These few aspects and many more made development with Java much easier than with most languages before [Sch02].

The main difference between Java and most other languages, however, is that the Java compiler does not create an executable binary file tailored to one CPU and one specific operating system. Instead, a common bytecode file is constructed that can only be used in combination with a so-called virtual machine, i.e., a runtime engine for Java applications. Obviously, these virtual machines have to be adapted to the different machines, but the bytecode remains the same. In other words, whenever a JVM (Java Virtual Machine) is available for a specific system, all Java applications can be used.

Unfortunately, until now, the vision behind Java did not come entirely true. The main edition of Java - called J2SE (Java 2 Standard Edition, replaced by J5SE just recently) - does indeed run on all currently available operating systems for desktop and notebook PCs. However, there are no official ports of the virtual machine available for PDAs, mobile phones or all the other devices envisioned earlier. One reason for this is the immense extent of the library: the core class library of J5ME has a size of 35 MBs. This, of course, is too much for smaller devices, where applications have to be kept as small as possible.

Instead, a subversion was brought to existence, called J2ME - Java 2 Mobile Edition [KH02]. This version features a very small but efficient library set, specific methods to deal with tiny screen sizes and low CPU power and a high compatibility with almost any programmable device available. The main focus, though, is on the smallest of devices, mobile phones for example, so PDAs, getting more and more powerful, are omitted again. Furthermore, it is not possible to create an application that runs equally well on both Standard Edition and Mobile Edition.

Given that modern PocketPCs can be as powerful in respect of CPU power, main memory and storage as the usual desktop PC in 2001 was when J2SE was first released, one can assume, that there will be a "real" port of Java Standard Edition for PocketPC anytime soon (although it is not announced yet). In the meantime, a viable alternative are several JVMs available for PocketPC and Palm that use an older specification called "PersonalJava". With these JVMs it is at least possible to run those applications that are compatible with an early version of Java (version 1.1.8).

3.2 Related Work

Almost as soon as the first mobile pocket sized computers hit the market, several teams world wide investigated possibilities to use them in educational scenarios or collaborations. Of course, not all projects aimed at improving interactivity in classrooms or other large meetings. Many tried to facilitate other activities, like note taking, floor control or easy access to central computers instead: Classroom 2000

[AAB+98, AAF+96], NotePals [DLB+98, DLC+99], SharedNotes [GBL99] and the Pick-And-Drop paradigm [Rek97, Rek98], to name but a few.

In the following two sections, we will concentrate on concepts based on improving interactivity and thus present a representative selection of projects supporting interactive communication. After that we will identify the shortcomings of these systems, explain the features such a system should have to be usable in a wide range of scenarios and present our own approach in this context.

3.2.1 Early Projects

Wireless LAN (IEEE 802.11) and Bluetooth are still very new technologies, so the pioneering projects in the field of pervasive computing in education and collaboration often used other means of wireless communication.

One of the very first projects in this area is *ClassTalk*, a system published by Better Education, Inc. [DGL+96]. Originating in 1994, ClassTalk was the first system that allowed a real-time analysis of the results of small quizzes during a lecture or class. The multiple choice question was displayed using a data projector, usually four answers were defined. The students had to work out the answer in small groups of three or four, then send the answer to one or more central nodes; lacking modern wireless communication, infrared transmission was used instead. A device that was programmable and included an IR port at that time was the Texas Instruments graphical calculator: the students only had to press one of the four buttons "A" to "D" to submit their answer. All answers collected, a bar graph with the accumulated results was displayed on the projector. Later, the software was refined considerably, allowing the teacher to analyse the answers more thoroughly and identify individual students and their answers. Also the calculators were replaced by proprietary devices which resembled very simple remote controls. In 2000, Better Education, Inc. ceased to develop ClassTalk that is still used in many North American high schools and colleges.

Another project that identified the need for computer supported interaction in classrooms very early is the *Pebbles* project of Carnegon Mellon University [MSG98,

Mye01]. Pebbles is a very large project, trying to investigate all possible uses in which mobile devices can interact wirelessly with stationary PCs. One of the more famous software products emerging from Pebbles is the Remote Commander [MSG98] that can be used to control mouse and keyboard of a PC with a special software running on a PDA, and the SlideShow Commander that turns a PDA into a PowerPoint controlling device. For classroom scenarios, a special piece of software was created that - unfortunately - was never released or improved, although it was tested in the Spring 2000 in a chemistry class at Carnegon Mellon University with some success. 120 students were equipped with PocketPCs (HP Jornada), which they used to answer short multiple choice quizzes. Depending on the results of a test, a topic was recapitulated or skipped, so the students could actively change the course of the lecture [CMD00]. Pebbles was also the first project using wireless LAN in large lectures.

3.2.2 Current Projects

The application of wireless mobile devices in lectures has become a well examined field of study over the last few years. Although the main idea may be very simple, many different concepts have evolved

Very interesting features are offered by *ClassInHand*, a project at the Wake Forest University in North Carolina [WFU05]. All devices in this scenario are PocketPCs, one of them is turned into a web server, a presentation controller and a quiz and feedback device for the lecturer. With this handy unit, the lecturer can do several things at once:

- control the presentation, i.e., go to the next slide, read remarks on the PDA's screen or go back to an earlier slide,
- offer various material for the students to download,
- observe a simple feedback given by the students (content - not content),
- ask some multiple-choice questions about the current subject and analyse the results and
- gather textual feedback or questions entered by the students.

The students only need to have a web browser installed to access feedback, quiz or additional materials, so most devices can be used without having to install any software.

ConcertStudeo, a project of the Fraunhofer Institute for Integrated Publications and Information Systems (IPSI) in Germany goes a step further in developing a complete, commercial package for interactive teaching [DDFW03a, DDFW03b]. The basis is an electronic blackboard that is compatible with PowerPoint but offers a number of additional features for the teacher to improve communication between him or her and the students. The students use PocketPCs and a special software to access several interactive events like quizzes, collaborative brainstorming (which can be seen in Figure 3.4) or small role-playing games or to give a feedback to the teacher. ConcertStudeo aims at smaller groups of about 30 to 40 students.

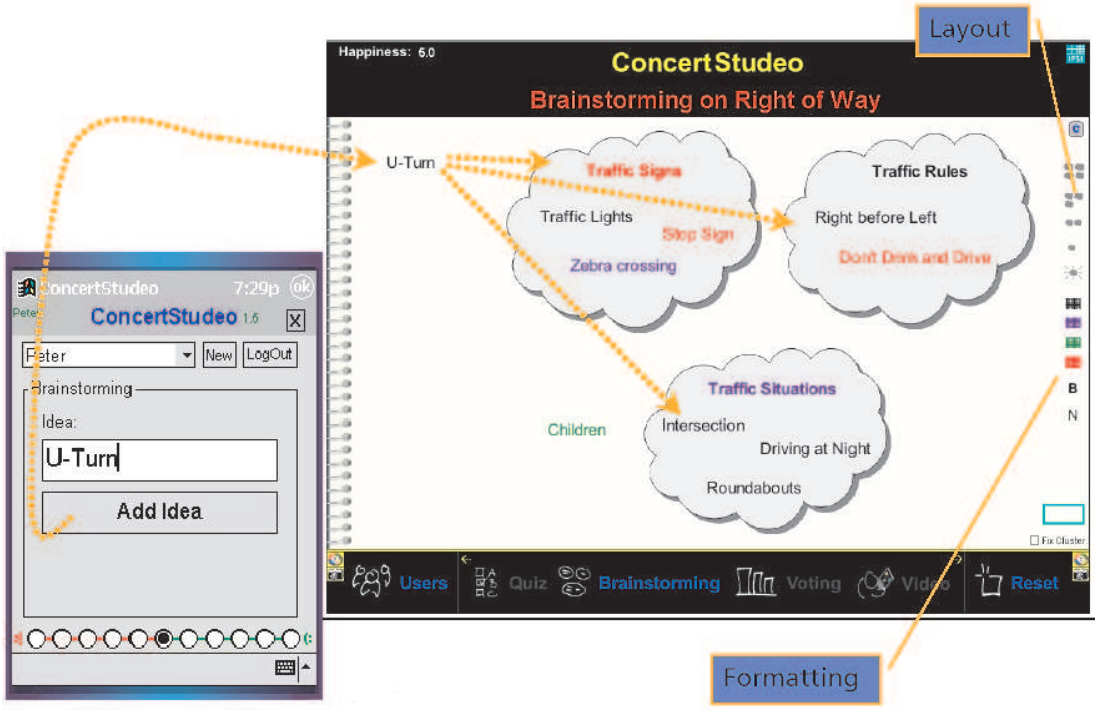

Figure 3.4: The collaborative brainstorming tool from the ConcertStudeo suite: students' and teacher's view

Specifically designed for online feedback is *CFS*, the Classroom Feedback System from the University of Washington [AAV+03]. It allows students to post annotations directly on lecture slides: the students use their notebook PCs to post the feedback

by clicking at a location on a slide and selecting a category from a fixed menu (e.g., "more explanation", "got it", "example required"). The teacher's view shows the number of feedback requests for each slide in an overview; on the current presentation slide the aggregated feedback is displayed with shaded dots of different colours at the denoted locations (one colour for each category). No names are displayed, however, to guarantee anonymity. Of course, the publicly projected slides do not show these markings, neither do the students' devices. At the appropriate moment, the teacher can respond to the received feedback.

Discourse is a fairly new commercial package of tools for interactive lessons in K-12 published by ETS [ML03]. Each student is supposed to have his or her own computer in this scenario. Since discourse operates Web-based, almost any type of computer may be used, as long as it is able to connect to the server and has a web browser installed. Using their computers, students can answer quizzes of various kinds (multiple choice, fill-in, voting, etc.), give textual feedback or send questions to the teacher. The design focus for the teacher's software, in contrast to most other systems, is to allow him or her to monitor each individual student and his or her success. Only basic accumulation or analysis of the data is implemented, instead very powerful report generators for assessments and all other activities are provided. Screenshots from the Discourse teacher software can be seen in Figure 3.5.

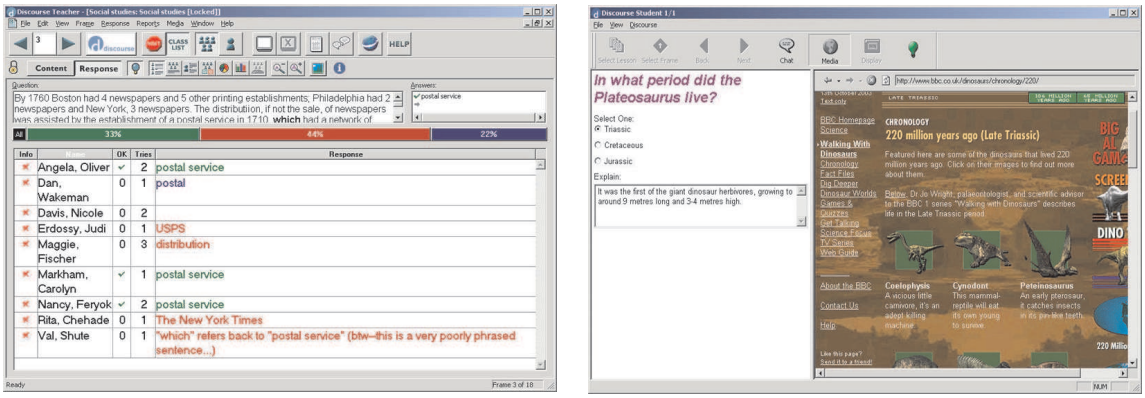

Figure 3.5: Discourse: quiz with web support (left), detailed assessment report (right)

Another commercial package, the *Classroom Performance System* (short CPS) released by eInstruction in 2000 continues the legacy of ClassTalk [eIN05]. With a set of proprietary devices very similar to remote controls, students in K-12 or higher edu-

cation can answer multiple-choice or numerical questions. The answers of up to 2000 individuals are sent by means of infrared to the teacher's computer using a supposedly secure protocol. In order to make the setting more interesting, several modes for the quizzes are provided, ranging from competing quiz games to official exam assessments. All data collected and analysed this way may be uploaded automatically to a central Internet server offering different levels of anonymisation and aggregation. For example, principals can monitor the status of all students in their school and compare the results with other schools, teachers can look into last year's accumulated grades of the class they are now teaching, and parents can see the (assuredly unaltered) grades of their child.

3.2.3 Our Approach

While investigating the different existing solutions for an interactive lecture, we discovered a number of common flaws. The first shortcoming is that most of the projects offer just a unidirectional flow of communication from the students to the teacher using the wireless network. The lecturer's only way to communicate with the students is with traditional means. Therefore, it is not possible to give each student an individual feedback (e.g., an analysis of his or her answers to the last quiz or the answer to his or her question). Furthermore, this narrows the possibilities of such a system severely: There is no way to adapt or personalise the additional interactive programme to the individual student's needs or to offer interactive tasks where the students have to exchange data amongst themselves or where the solution of a task depends on continuous communication with a server.

The second flaw is that the software used in most of the projects is very static. It offers a certain number of different interactions (i.e., quiz, feedback, ...), but it is not possible to extend the functionality in any way. To add new communication schemes would therefore require to contact the authors of the project or to use two different solutions simultaneously which is both unwieldy and error-prone. In some projects it is even impossible to deactivate some of the functionality if the lecturer does not need it or thinks that it may even have a negative effect on his or her style of lecturing.

Last, but not least, there is a strong tendency to favour K-12 scenarios or small working groups in many of the presented solutions; this means that the target class size is about 20 to 30 students. In larger scenarios (lectures often have more than 150 students) the solution is often hardly applicable, because too many proprietary hardware devices had to be bought or the software lacks adequate analytical methods to help the lecturer comprehend all of the incoming data.

Another minor problem with many software projects is the dependency on a certain computer platform for the teacher client software and - in some cases - for the students as well. Even when the students only use a standard Web-browser to access the service, the user interface is in many cases not suitable for PocketPCs or other devices with small screen size.

Out of these considerations, we deduced the following set of features an interaction-enhancing software system should have to be suitable for large classroom scenarios:

- Continuous 1-to-n-to-1 communication, allowing an uninterruptable transfer of data from the students to the teacher, as well as a controlled data distribution from the teacher to single students, all students at once or to a specific group of students.

- A modular, easily expandable architecture. On top of a monolithic server core any number of modules, each containing the logic of a certain interactive communication scheme, can be loaded. To support the development of new modules, a sophisticated programming interface is provided by the software.

- Scalability in terms of the number of students connected to the teacher. The software should be equally well applicable in K-12 as it is in large lectures. Multiple abstraction layers should provide the teacher with different levels of detail concerning the processed data; the lecturer can then choose by him- or herself, which abstraction is most suitable for a given class size.

- Platform-independent software on standard hardware should be used. Proprietary devices are usually expensive and lead to an unacceptable dependency on the manufacturer. Software solutions, on the other hand, are more easily accepted, the better they perform on any given computer platform.

Regarding this list of important features, we developed a new software solution called WIL/MA which is described in detail in the following two chapters.

4 Architecture and Design Principles

After an overview of the general architecture of the software toolkit presented in this dissertation, important parts are discussed in detail: first, the main components of the server software, then the modular structure of the toolkit and finally the design principles for the client software. Clustering functionality and connectivity improvements are described in the final part of this chapter.

4.1 General Architecture

The interactive lecture attempts to create additional channels of communication between the teacher and the students. Each student has his or her own mobile device with which he or she can send data and receive data. It is not primarily intended for the students to send data amongst each other, however, this kind of interaction should also be possible, though only in a controllable fashion. But not only in respect to the flow of communication the teacher is the focal point. He or she also has to have the total overview of all collected data while access to most detail levels of information should be deliberately restricted for the students' satellite clients.

These reasons led to the decision to design the software toolkit as a classical server/client architecture: a central server (usually running on the machine of the teacher) accepts and administers connections from any number of clients. All data flow can be controlled perfectly well in this architecture, and the server is by definition a barycentre of information. Also, this concept allows a very reliable communication between all participants since there is a central instance that can easily discover problems and enact counter measures very quickly. Storing all data at a central point also has the advantage that little or no information is lost should one or more clients

become disconnected from the pool - an incident not too improbable in a scenario that completely relies on wireless communication. Of course, one might assume that a central server might represent a performance bottleneck, but our experience shows that more than 300 clients can easily be handled with an average notebook as server (see 6.2.1 on page 104).

4.2 Server Components

The main part of the server in this architecture - the core - is void of any logic concerning the different ways to improve interactivity in the lecture. The main purpose is to gather connections from different clients and manage the communication and the data flow in a controllable way. All other functionality is sourced out to a number of service modules that are loaded by the core of the server (the server skeleton) at start-up, each module offering one specific service to be used in the interactive lecture. An overview of the architecture and the classroom configuration can be seen in Figure 4.1 on the facing page.

Now we consider firstly the basic components of the server skeleton.

4.2.1 Connection Manager

The *connection manager* accepts connections from clients and checks for their validity. The latter is done with a short handshake protocol in the initial phase, which also includes the exchange of public keys for encryption and a list of available interactive services (see chapter 5.3.2 on page 65). All allowed connections are stored in a list that is continuously screened for latency problems or lost links.

The more important parts of the connection manager, though, are a message distribution system and a message queue. The message distribution system is just like a post office; all messages created inside the server complex or routed through the server are provided with an address stamp identifying one or more recipients with name, access level, and/or origin. Dependent on this information, the messages are multiplied where necessary and distributed to the message queues of the individual

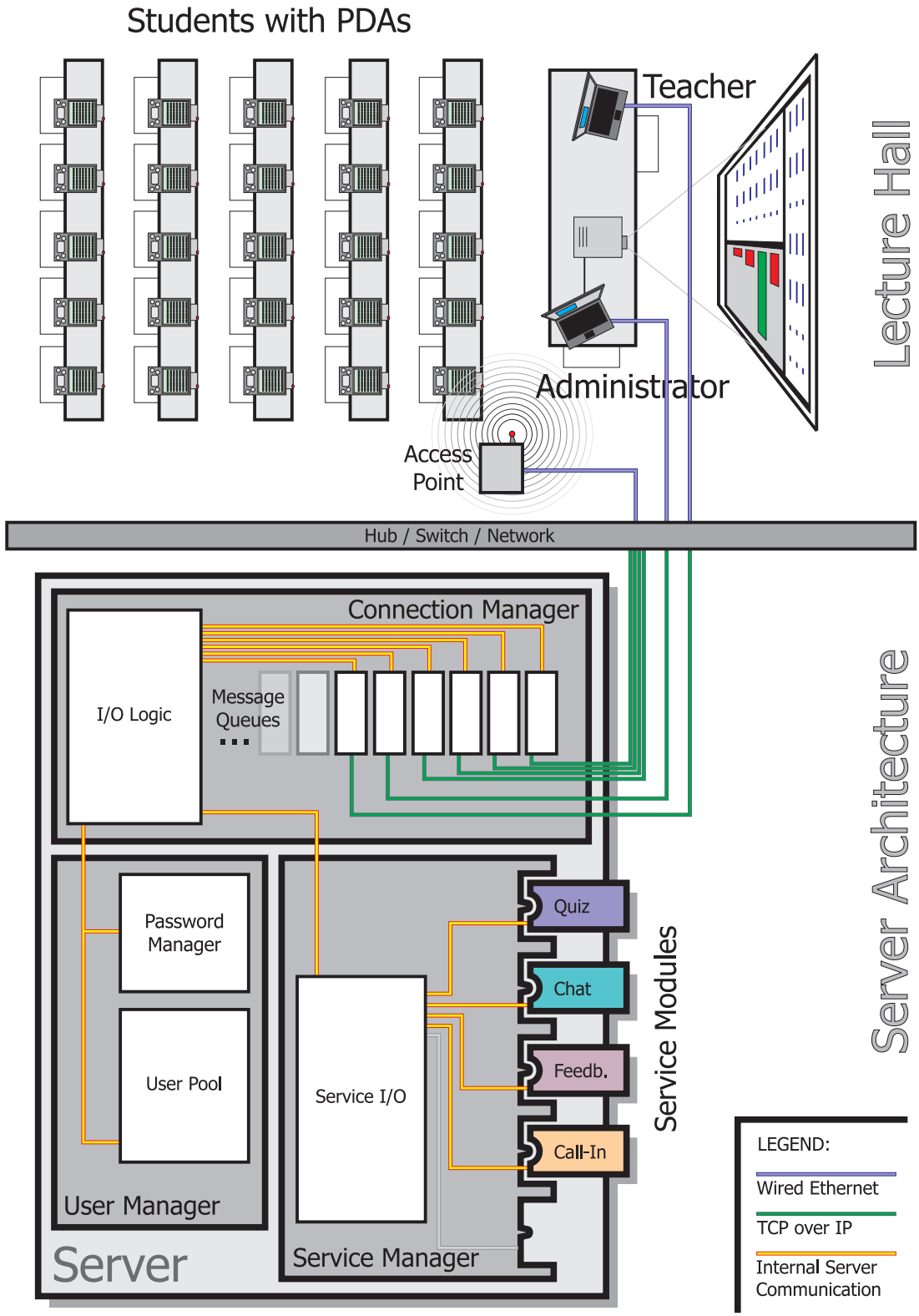

Figure 4.1: The architecture of the WIL/MA software: Communication inside the server (bottom) and communication in the class room (top)

addressees. Each connection is assigned its own queue where all messages intended for that connection are stored and sent out as soon as possible.

If the server and the clients are configured accordingly, the message distribution automatically switches to a reliable multicast protocol if a message exceeds a specified length and many users are supposed to receive the message. This considerably reduces the network load when broadcasting larger amounts of data, e.g., pictures or lecture slides. A detailed description of this technique will follow in chapter 5.3.3 on page 70.

4.2.2 User Manager

A connection is always assigned to a single user identified by the server. This identification process is done by the *user management* in two steps. First, during the initial handshake phase, a username is expected from the client. This username may or may not match an already known user. In the second step, the user has two possibilities: he or she can log in, providing a password that is stored for his or her username in the internal list of the user management. This, of course, requires the user to be registered with the system. If the password provided is not correct or if the username is not yet registered, the login will fail, and the connection will be cancelled. Alternatively, the user can register with the system and store his or her username along with a new, freely selectable password. This will fail if the username is already registered to another person. When the user has successfully registered, he is granted a default set of access rights that can be configured by the server administrator.

After a user is properly logged into the server, he or she is assigned a set of access rights that is stored with the username in an extensive list administered by the user management. Access rights are divided into general access and service access. General access determines the rights of the user regarding key functionality provided by the server skeleton, e.g., removing individual users from the user list, changing the access rights of other users or adjusting basic parameters. The service access determines the privileges of individual students in the respective services. Per default, all service access levels align with the general access of a user but they can be overruled on an individual basis at any time. This way, certain users, for example, can be denied the

access to a specific service method while they still have full student access to all other services.

All user information is stored in an XML database loaded by the server at start-up and updated whenever something changes. This list not only contains username, password and access rights, but also any additional information that has to be available to the system or a service the next time the server is started.

4.2.3 Service and Configuration Manager

As the name already implies, this server component has two duties. The first is to load a textual list of configuration parameters at start-up that can be used by the server administrator to change the default behaviour of the server. The parameters are stored in a list from which they can be retrieved by corresponding methods of the server skeleton or any service.

The second duty is to manage a number of services that are loaded during start-up by the server. Each service is assigned a unique identifier code with which it can be addressed by the clients; a list of all available services with their ID is sent to each client during the initial handshake phase by the connection manager (see chapter 5.3.2 on page 65). At runtime, the manager acts as an interface between the service modules and the server skeleton, offering a number of valuable methods for the services, i.e., the routing of incoming messages from the clients to the addressed services and the automatic marking of outgoing messages with the proper ID.

The Service Manager is also responsible for access control in the service modules. To access a certain service, the user has to register to it after he or she has logged in; this is usually done automatically by the client software. Depending on the access granted to the user and the configuration of the service, the user's registration succeeds or fails. Once registered, the user's client can send message packets addressed to the service module. To avoid misuse, the packets are checked for integrity first before they are forwarded to the appropriate module along with the username of the sender and his or her access rights.

4.3 Services and Service Modules

The peripheral part of the server is made up of any number of independent service modules loaded by the server skeleton, each implementing a certain interactive service to be used in the lecture. An interactive service in this context is a moderated channel of communication between the teacher and the students. For example, it may be a quiz service, i.e., an application to send multiple choice questions to the students, collect the answers and display an analysis of the results. Other exemplary interactive services offer the possibility to give feedback or to pose a question at any time; for a detailed discussion of currently implemented services[1] see chapter 5.6 on page 87.

A service module has to satisfy two requirements. First, the communication between the service module and the clients has to be regulated. Second, incoming data has to be accounted to be able to provide different levels of aggregation. Both parts, though interwoven, have to be considered individually when implementing a service module.

4.3.1 Interaction Rules for Services

The service module must implement all the interactions needed for the interactive service, i.e., the rules for communication between the clients and server. This unfortunately very error prone part is basically comprised of a set of commands a client can send to the service and a set of messages that are sent back as answers or update information packages. To ease development, the server core enforces several basic principles concerning the format of commands and responses.

Commands to the service module have a unique numerical code that determines the method to be called. The parameter set of each command can be defined solely by the service without any restriction. It is encapsulated into a closed package, so a misinterpreted command can not mess up the subsequent communication. Additional information supplied by the server manager includes the calling user's name and his or her access rights. Each command must be answered with a message to the client,

[1] as of January 2005

even if this message is left empty. This way, the client can make sure if its command was processed.

All messages that are sent to the clients must contain two pieces of information. First, the ID of the source module and second, an alphanumerical identification code for the type of information contained in the message. Both are needed by the client to handle the message properly, as - again - the format of the remainder of the message is completely arbitrary. Usually, there are many more different message types defined for a service than there are commands. This is due to the fact that a single command often results in two or more different responses ("command was processed successfully", "error xyz occurred" etc.) and may additionally lead to update messages sent to the teacher or all other clients or both.

4.3.2 Aggregation of Incoming Data

Being one of the most crucial of all design principles, the service module is required to aggregate, analyse and store all incoming information automatically; otherwise it would be impossible for the lecturer to handle the vast amount of data that may come up during a common lecture. Instead, the teacher as well as the students are provided with only top-level (i.e., highly aggregated) data by default; methods to retrieve more detailed information are provided.

4.4 Clients

In an interactive lecture, there are at least two different types of participants: the student and the teacher. A third type may be identified as an assistant, helping the teacher handle the interactive services. Each type of user has different requirements with respect to the display, management and amount of data gathered by the service modules:

- The *student* is generally only interested in data concerning him or her. Instead of providing large amounts of data, it is more important to prepare selected data to be conceived quickly and easily, so that the student can still concentrate on

the lecture. For the same reason it is necessary for the student to be able to enter commands sent to the server effortlessly. There is little need for the student to access any information gathered during a lecture afterwards or preparing input ahead of time, so an offline mode is not required. Also, when designing the software for the students, it has to be kept in mind that the display of the device typically used in the interactive lecture is rather small.

- The person controlling the interactive part of the lecture, henceforth called *administrator* (it may be the lecturer himself or an assistant), has almost completely different interests. Beginning with top-level aggregation, the interface for the administrator must be able to display all possible detail levels of data, sometimes even in different views. Fortunately, screen size is no issue since the typical device to control the interactive lecture is a notebook or desktop PC. Also, in contrast to the student, the administrator needs to prepare some material for the lecture before it starts, and also may want to analyse some of the collected data after the lecture. Both requirements demand for some service-related logic to remain in the client.

- If the lecturer has an *assistant* to manage the interactive part of the lecture, the optimal interface for him or her provides only the most basic information and as few control options as possible. The ideal client is kept small and unobtrusive, hovering in a corner of the screen only to be used when the lecturer has time to do so or if the lecture requires the application of an interactive service. Information gathered from the students should be suppressed except for critical data (e.g., if the feedback of the students is notably less than optimal or if there are remarkably interesting questions in the incoming queue) but accessible at any time.

This divergence in requirements of the different client types led to a couple of specific design principles the complete software kit is built upon. The most obvious consequence, of course, is to develop at least two different clients: one for the students and one for the administrator. This is easily possible due to the tiered access right management of the server core and the most reasonable way to provide the target user groups with a well-structured application most suitable to meet all their needs.

Second, the server and the modules for the interactive services have no integral interface for supervision; the administrator client is needed to do that. This way, the administration is independent of the server; it is possible to run several administrator clients at the same time and - more importantly - to run the client without a server to create data offline that is later to be used in the lecture. The administrator client is therefore no dumb terminal, but contains most of the logic of the services.

Third, all data gathered and analysed during a session is stored on the computer running the administrator client and not on the server computer. Again, the main reason for this is the need for offline editing, but it also emphasises the independence between administrator and server. A single server can be used by several teachers this way, and it is more difficult to get access to possibly classified data when it is stored securely on the personal notebook of the lecturer. Unfortunately, this principle interferes with the possibility to run more than one administrator client simultaneously, so specific mechanisms may be required to guarantee the integrity of critical data (e.g., using different access levels inside the administrator group).

Last, but not least, the different client applications have to be built in a way similar to the server software. A basic framework controls the communication with the server and provides extensive methods for a uniform user interface. This scaffold is then completed by loading several modules containing the client logic for the distinct services. After booting the client and logging in, the user can then switch between the services using a menu or a similar control method. In contrast to the server, however, the services to be loaded are not determined by the user but forced by the server software.

4.5 Connectivity and Clustering

Usually it is sufficient to use one server for an almost arbitrary number of clients in an interactive lecture. There are, however, settings where the utilisation of only a single server may not be optimal:

- Tele-lectures, i.e., the broadcasting of a traditional lecture to remote locations, are becoming more and more common. Particularly for the students in the

receiving lecture halls it is interesting to use the interactive services because they are even less able to interact traditionally with the lecturer than the local students. The server, of course, will be a machine in the lecture hall of the sending university, so in order to access the server and thus the interactive lecture, the students have to have access to the Internet. This is usually a feature that is disliked by most of the lecturers because it tends to be an irresistible diversion for the students. So either a special countermeasure has to be installed (e.g., a firewall) or a second server that synchronises with the main server.

- There are lectures or other meetings with more than 300 participants. When the lecturer wants to use bandwidth- and resource-intensive services, the resulting load may be too much for a single server. A solution is clustering, i.e., using several servers connected to each other and distributing the connections equally among them.

- Home learners, i.e., students attending a lecture via the Internet from any location, usually use special software designed to display the lecture slides with the annotations of the teacher and to play the voice of the lecture. These shared whiteboards often feature interaction models like virtual handraising or feedback, which are very similar to the services of an interactive lecture (a good example for such an electronic whiteboard is the mlb - multimedia lecture board [VM01]). In this case, it would be feasible to exchange the information gathered by the whiteboards with that of the interactive lecture.

Therefore, the server software provides specific methods to enable connections between two servers: each server is started individually and remains independent, but allows synchronisation with other systems by means of interface programs. These interface client applications connect to two servers using a special username with additional rights “synchronisation user”. Once an inter-server connection has been established, the servers exchange all data necessary so that all servers have a common data pool afterwards. Each following action induced by one of the clients (including interface clients!) that changes the data pool of one server causes update messages to be sent to all other connected servers. This way, small networks of interconnected servers can be assembled. As a further advantage, the data exchanged between the

servers is preaggregated by each individual server and thus needs significantly less bandwidth than the accumulated data rates of the individual clients.

Instead of connecting two servers, it is also possible to connect a server to a foreign software system, provided that it has sufficient methods to be able to synchronise data. The interface then has to translate the data packets between the two systems. An exemplary interface client has been implemented to connect a server to the mlb.

5 Implementation

Beginning with an overview over the WIL/MA toolkit and its different packages, this chapter will deliver deeper insight into the implementation issues: Communication and messaging as one of the most fundamental parts of this software, design of user interfaces for the students, dispatching tools to ease the configuration of interactive lectures, and finally the three interactive services that were implemented and used in numerous studies.

5.1 The WIL/MA Toolkit

The software toolkit presented in this dissertation was after several - for various reasons - unsuccessful attempts of finding a proper name finally baptised *"WIL/MA"* which stands for "Wireless Interactive Lectures in Mannheim"[1]. The source code is about 820 KByte in size and the 170 classes sum up to about 30,000 lines of code (comments and legal headers not included). It is an almost completely rewritten version of an earlier prototype (the *UCE toolkit*) that was used in the first two evaluations [SME+02]. A short history of all releases of WIL/MA and its predecessors is shown in Table 5.1 on the next page.

This most current version is published under the GPL (GNU Public Licence, [GNU05]) since 2002. Everything is written from scratch in Java, which is a royalty-free language, so there are no hidden license restrictions. The only external package

[1] The first name: "UCE" (Ubiquitous Computing in Education) was replaced by "WILD" (Wireless Interactive Learning Devices) with the release of the first public version; many older publications refer to this name. Unfortunately, "WILD" was already registered by a German company, so "WIL/MA" was chosen as the final label.

Date	Name	version	description
Feb 2001	UCE-Tools	0.1	Initial release of a prototype featuring multiple choice quizzes and feedback. No unified clients yet.
Nov 2001	UCE-Tools	0.3	Release including Call-In and multiple choice quizzes with an arbitrary number of correct answers.
Apr 2002	UCE-Tools	0.31	Bug fix release for the first evaluation
Jan 2003	WILD	0.5	First prerelease version of the new design
Apr 2003	WILD	0.7	Release for the second evaluation including an improved client-server communication and an unified student client with new design. Support for quizzes and online-feedback.
Jun 2003	*CodeBreaker*	*1.0*	*Special release for the CodeBreaker curriculum, see chapter 6.4.2 on page 122*
Sep 2003	WILD	0.8	Release for the third evaluation with many enhancements for an easy setup, improved reliability and user friendliness
Apr 2004	WILD	1.0	First public release, featuring two unified clients for students and administrators. Support for Quiz including new quiz types, like “clickable images” (see chapter 5.6.1 on page 88), Feedback and Call-In.
Sep 2004	WIL/MA	1.2	Second public release with multicast support and several enhancements in the user interface API.

Table 5.1: History of the WIL/MA toolkit

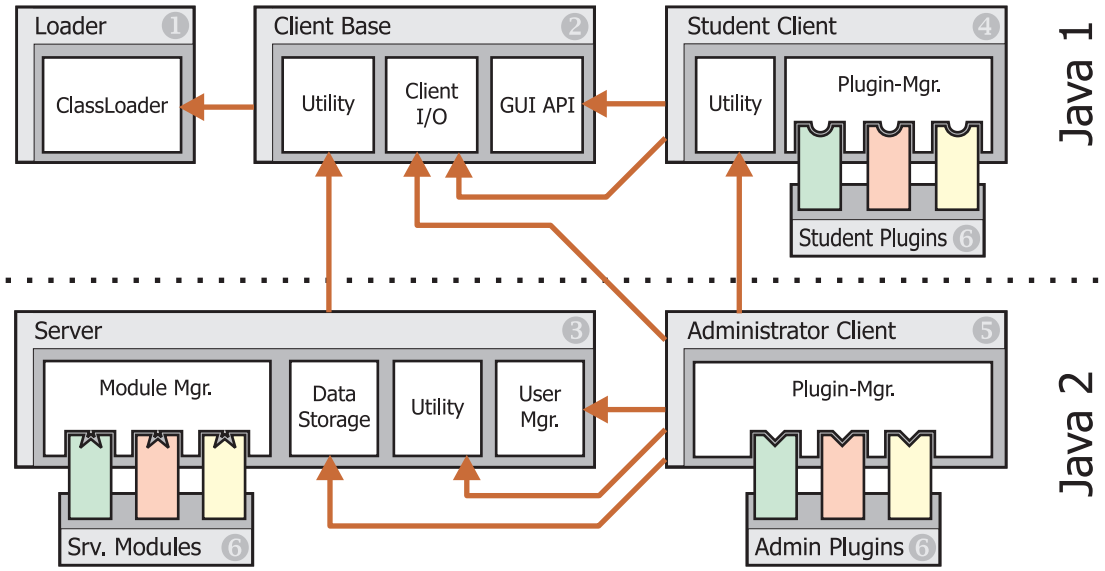

Figure 5.1: The packet structure and dependencies of the WIL/MA software

used is a cryptography library released by an open-source organisation called BouncyCastle, which is also released under the GPL [LOB05].

For the implementation, two different Java versions were used: Java 1.1.8 (short: Java 1) and Java 1.4.1 (short: Java 2). The first version of Java was discontinued in 2000, but it is still the best way to ensure maximum compatibility with browser plug-ins and non-standard devices (see chapter 3.1.3 on page 36). Java 2, on the other hand, that has just recently given way to Java 5, offers high performance and flexibility for advanced software projects. To be able to use different Java versions and to meet the requirements of the different devices, the software compound is divided into six major parts (an outline of the package structure and the dependencies is shown in Figure 5.1):

1. The *client base* package includes all classes needed for communication between server and clients. It also defines a number of utility classes needed by all other parts of the software. This package is the most basic of all and as such written consistently in Java 1.

2. The *client loader* package is an addition to the client base package, providing a minimal subset of classes written in Java 1 needed to load or update the rest

of the software over the Internet from the server. This mechanism is described later in chapter 5.5.3 on page 86.

3. The *server* package is made up of all classes required for the server software, divided into the three management layers described in chapter 4.2 on page 46. Except for a small footage of core classes in the client base package required for communication with the clients, the server is built on Java 2, thus being very efficient in terms of CPU and memory usage. This package also includes a small number of utility classes implemented in Java 2 for use by the modules and other parts of the software.

4. The *student client* package provides the classes for the students' client software. This includes communication, a framework for different service plug-ins and a unified user interface specifically designed for smaller screens and stylus input (see chapter 5.4 on page 81). It is entirely built upon the client base and client loader package and uses Java 1 for compatibility reasons.

5. The *administrator client* package also relies on the client base package, but is itself implemented in Java 2. The administrator client offers full functionality needed by the teacher or assistant to manage the interactive lecture.

6. The service modules, although in fact being part of the other package structures, represent a sixth package. Each module is made up of three different parts:

   a) A server module, written in Java 2 and based upon the interface structure defined by the server. This part includes most of the logic to aggregate and analyse incoming data as well as to communicate properly with the clients. It also defines the data structures in which all the relevant data is stored.

   b) A student client plug-in, written in Java 1, which implements the interfaces given by the student client. The plug-in has to translate the user input into commands for its server module counterpart and interpret the responses sent back by the server. Its duty is also to display incoming data properly and to provide an intuitive user interface based on the strict generic GUI structure of the student client software. Since all data in a student client

is transient, the student client plug-in uses its own simplified data storage structure.

c) An administrator client plug-in that is written in Java 2. Like the student client plug-in, it has to be an interface between a user (the administrator in this case) and the server. Since the requirements of the administrator are considerably higher than those of a student, the user interface has to be much more complex. For this reason (and due to the fact that the screen size doesn't matter), the administrator client is less restrictive in terms of a unified GUI. The plug-in uses many of the server module classes, particularly those for the data storage structure, to be able to offer the required offline mode.

The following sections will cover a selection of interesting design and implementation issues concerning the WIL/MA software. A user documentation as well as the API specification is included with the release package on the WIL/MA website [LL05].

5.2 Data Storage and Management

The basic idea of each interactive service is to collect certain information during a lecture and to use data that was either gathered in earlier lectures or prepared ahead of time. The data management is thus a very crucial part of the software design.

Data management takes place both in the server and in the administrator client. Both parts of the software need to have access to a common part of the information pool when a lecture is running. The only difference is the lifetime of the information: while data in the server is transient, the administrator client is responsible for storing most of the information on the hard disk or other non-volatile media. Due to the analogy of the required data pools of server and administrator client, both parts of the software use the same classes to store their data.

These class packages have to fulfil a number of duties besides storing the data in an efficient way. First of all, the storage classes have to contain most of the logic needed for aggregation and analysis, so that not only the server, but also the administrator

client can use these methods at any time; this is due to the design principle explained in chapter 4.4 on page 52.

Second, the storage classes provide methods to pack all permanent data into XML compatible documents and to restore the data objects from such a document. A static class, called XMLFactory, can be used to do the transformations. Two methods - "saveElementToFile" and "saveElementToString" - perform the packaging of data objects which are instances of classes implementing a certain interface: XMLElement. Beginning with the specified root element, the objects belonging to the data pool are prompted to store their data into a given XML document encircled by a tag carrying the same name as the corresponding Java class, thus preserving the tree-like class structure. The XML document is then saved to disk or returned as a string. The storage classes are relatively free in respect to the way they store their data, as long as the structure is maintained, thus achieving a high flexibility needed to meet the requirements of the usually extremely different services.

The XMLFactory can also be used to restore the data. To do this, the storage process is simply reversed: The complete structure preserved in an XML document loaded from disk or provided as String is transformed into objects of the corresponding classes. If the name of a tag is known to the system as a storage class, the object is automatically reconstructed by calling the appropriate method of the class. Data stored in other tags has to be collected and retransformed by the classes. During the transformation, it is made sure that the original tree structure is recovered.

The methods of the XMLFactory are an easy way to store and restore all prepared or gathered information to disk. It is even possible to transform only a part of the tree to be exported to another data set or to create perfect duplicates of selected branches. To support the implementation of the save and restore methods of a service, two common elementary elements are implemented: a text element with some text layout information and a picture element that can be used to store image data in the BASE64 format into an XML file[2].

[2]BASE64 is a transformation of an arbitrary 8 bit data stream into a code consisting of 64 different ASCII characters. All these characters can safely be used in almost all text-based formats or protocols.

Last, but not least, the interface for the data storage classes provides methods to create binary data streams as an additional representation of the data stored in the storage objects. These data streams can then be used as an easy way to provide the students' clients with vital information. Of course, neither do all storage classes have to implement these methods nor is it necessary to transform all the stored data into the data stream since the students' clients most often only require a very small amount of information.

5.3 Communication and Messaging

A huge part of the WIL/MA software is dedicated to the transport of messages. This part consists of a protocol, controlling the communication between server and clients, a message creation and broadcast system inside the server and a message distribution and command building system in the clients. Furthermore, it provides methods to broadcast large data chunks more efficiently to the audience, to encrypt and sign critical data and to synchronise data with other servers.

5.3.1 Considerations against HTTP and HTML

Many software solutions that had been developed for interactive lectures (see chapter 3.2 on page 36) use the well-known protocol HTTP for the communication between server and clients. In combination with the markup language HTML this has the huge advantage that a standard Web browser on the client device is sufficient to access the extended interaction schemes that are provided by the software. Since almost every operating system for any kind of device - including mobile phones - provides a built-in web browser, this may seem as the perfect solution: The setup is very easy, since no software has to be installed on the students' devices and almost any network-capable device may be used. Furthermore, the user interface on the clients is well known to the majority of students that have experience with the WWW.

However, using HTTP and HTML has some severe disadvantages. First of all, the Web browsers installed on the different devices show huge differences in the way the

information on a Web page is displayed. This, of course, complicates the effort to create a consistent design that looks equally well on all devices. Furthermore, active components, like check boxes, buttons, text fields, etc., usually take up too much space especially on computers with small screen size, which makes it very hard to maintain a concise user interface.

Another drawback is that the Web pages used in this context have to be static. Although the Web browsers on notebooks or desktop PCs offer a wide range of dynamic functionality (e.g., JavaScript, Java, Flash animations), this is not supported by the browsers installed on PocketPCs or smaller devices. Of course, this limits the possibilities of the software to a very simple data exchange between server and client; it is not possible to create services with which the students can interact for some time before sending the resulting data to the server (e.g., interactive simulations). Furthermore, the Web pages have to be delivered by the server as a whole; it is not possible to update just a part of the page.

This leads to a problem of the HTTP protocol when used in this scenario. The design of HTTP allows for a client to retrieve certain information from a server; after this information is received, the connection to the server is terminated. It especially does not support the server-triggered broadcast of any kind of information to a client. On the one hand, this means, that it is impossible to detect if a certain student is still logged in (he or she may just not be interested in participating right now). On the other hand, it is very hard to create the 1-to-n-to-1 communication we see as a most important feature in such a scenario: The only way for a client to see whether the server has new information in store is to continuously reload the current Web page. Of course, this technique (called “polling”) wastes very much bandwidth since every time the server has to resend the complete Web page even if nothing has changed.

For these reasons, we decided to develop our own protocol and a dynamic, platform-independent client software that automatically adjusts to the properties (especially the screen size) of the student’s device. Additionally, features have been implemented to ease installation, updating and configuration of the client devices (see chapter 5.5 on page 84).

5.3.2 UCEP - The Protocol of WIL/MA

The protocol used in the WIL/MA software is called *UCEP* according to the name of the first prototype software kit; the abbreviation stands for Ubiquitous Computing in Education Protocol. For the communication, TCP is used, and the default port is 2904. UCEP is a binary protocol - currently in version 3.0[3] - that is using only four different packet types with strictly specified structures: two for sending data from the client to the server and two for the reverse direction. The only exception to this rule is the "handshake", taking place immediately after the client connects to the server.

The handshake part of the protocol first of all has to make sure that the client uses the correct protocol version. Furthermore, some critical data has to be exchanged; the complete procedure of the handshake is listed in Figure 5.2 on the following page. During the handshake it is possible that clients are rejected if a predefined maximum number of concurrent clients is exceeded or if another connection with the same username is already active.

When the handshake is completed, the protocol switches to the main communication mode. In this mode the client and the server can send data to each other at any time with the only difference that the packets of the client have to be answered with a response packet while the server packets do not have to be acknowledged by the client.

Figure 5.3 on the next page shows the structure of the command packet sent by the client to the server. The size of these packets is exactly 18 bytes if no additional data is included (type A). Type B packets with a data payload have at least 20 bytes, but otherwise no size limitation. The contents of the packet fields are specified as following:

- The *sID* (service ID) specifies the service the command is intended for. For this, the internal number of the service is used that has been transmitted to the client during the handshake. If the command is not meant for any service but for the server core, the sID is set to zero.

[3] as of January 2005

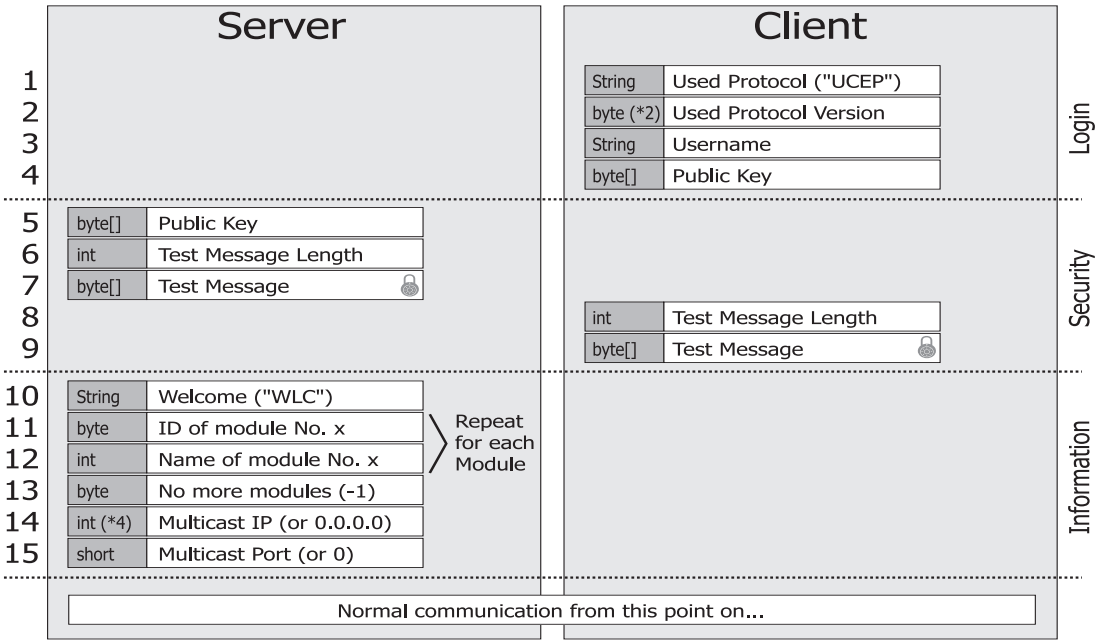

Figure 5.2: The handshake procedure of UCEP

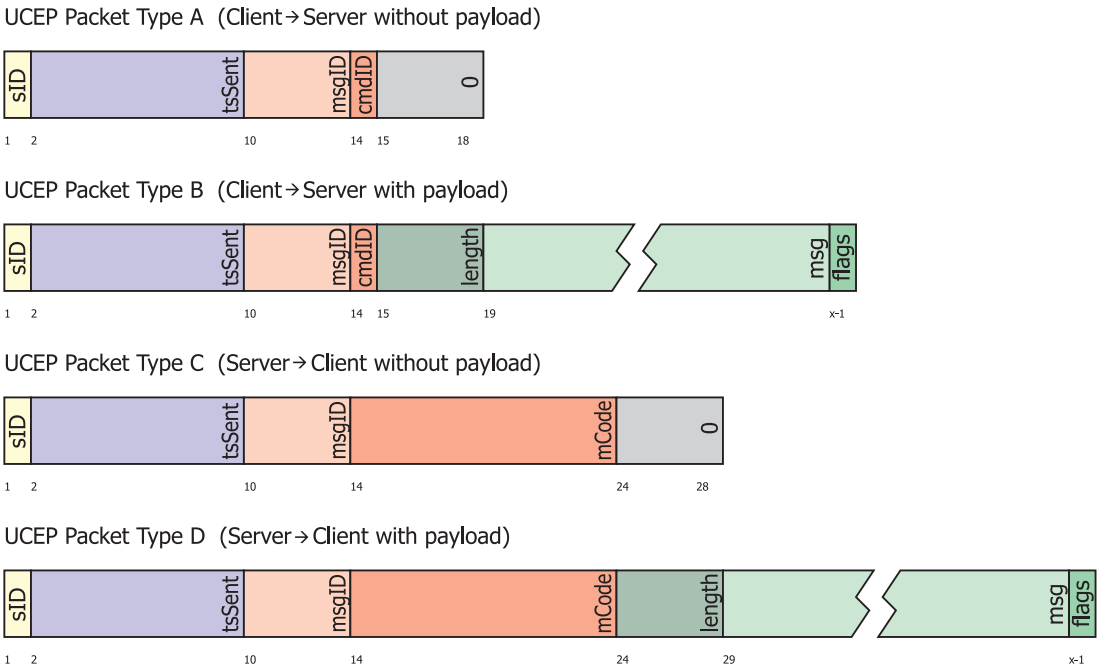

Figure 5.3: The structure of the four main UCEP packet types

- The *tsSent* value is a timestamp, i.e., the current time in milliseconds inserted automatically by the client base before transmitting the command to the server. This value is primarily intended for internal purposes, e.g., to detect communication problems or a beginning network overload. It may, of course, also be used by certain services to determine the timely transmission of time-critical data. Right after logging in, the communication class of the client base package performs a clock synchronisation with the server using the SNTP protocol [Pos82], thus this timestamp is quite accurate.

- The *msgID* (message ID) is a value whose main purpose is the supervision of the communication. This value - which is a simple serial number - is also automatically inserted by the communication class of the client core. After processing the command, the server will include the exact same number in the response packet. This way, the client can reliably match the responses to the commands.

- Most important, of course, is the next value, the *cmdID* (command ID). This byte specifies one of the eleven commands provided by the server core. Depending on this value, further actions are determined, e.g., performing a login or forwarding the command to the service specified by the sID. A complete list of all commands with a short description and the resulting actions can be found in Table 5.2 on page 69.

- The *msgLengh* value specifies the size in bytes of the payload included in the command packet. If it contains a zero, the packet is of type A and considered complete at this point. No further data will be read in this case.

- The *message* is actually a byte stream of the size specified by the msgLength value. The server will read the complete byte stream into a byte array without inspecting the contents; this is done later by the server core or (if applicable) the service module when the command is executed. Encapsulating payload data this way ensures the integrity of the communication stream at any time. Even if a service module is implemented incorrectly, e.g., by omitting to read a certain value, the remainder of the communication is not affected.

The payload may contain a number of predefined parameters for use by the called command. An error in the parameter set is handled by the command method in the server skeleton in this case. In most cases, however, the data payload is forwarded to a service module. If so, the first byte of the data stream usually is used to identify the desired subcommand defined by the service and the rest contains the information destined for that service. The responsibility for handling potential errors in the payload then switches to the service.

- The following byte (*flags*) contains several indications how the payload has to be handled. At the moment, only three flags are defined. The first two specify if the payload has been encrypted and/or signed. In any of these cases, the server automatically provides for the decryption of the payload or the verification of a signature. If an error occurs, i.e., the message could not be decrypted or the signature does not match, an error message is sent back in response and no further action is taken. The third flag marks the packet as multicast information packet, i.e., the actual payload is transmitted with multicast while the data in the flagged packet was replaced by control information. For more details on this feature, see chapter 5.3.3 on page 70.

As you can see in Figure 5.3 on page 66, the server response packets are very similar to the client's command packets. Again, there are two types for packets with payload (type D, at least 29 bytes) and without (type C, 27 bytes). There are, however, differences in the way some of these fields are used:

- The *sID* is automatically set with the number of the service where the message was generated. If it is a server core message, this value is zero.

- The cmdID was replaced with a *msgCode*. Instead of a single byte, a string with 9 characters in the format `xx-yyyyyy` or `xx!yyyyyy` is used, where `xx` is a two character code identifying the origin ("CS" for the core server, "QZ" for Quiz etc.) and `yyyyyy` describes very shortly the purpose of the packet (e.g., "STORED" would be an acknowledgement that some data has been stored, "NEWUSR" tells the client that a new user has logged in). Except for server messages, both parts of the code can be chosen freely by the service. The

code	internal	description
1	LON	Log into the server using a specified password
2	LOF	Log off properly
3	REG	Register to a specified (*sID*) service
4	UNR	Unregister from a specified (*sID*) service
5	SND	Send the attached data field to the service specified with the *sID*
6	SET	Set certain parameters like screen size, performance of the used client device, etc.
7	CMD	Server commands (close connections, update and verify data, get information about other users, etc.)
8	RGS	Used to synchronize data between independet servers
9	TIM	Synchronizes the time between server and client
100	PNG	Keep-alive feature of the UCEP protocol (see 5.3.2 on the following page)
101	MCS	Multicast support commands (see 5.3.3 on the next page)

Table 5.2: Server commands and their description

delimiter finally marks a packet as an error message (using the exclamation mark) or as a normal info or data message (using the hyphen).

The reason for this design is that there are far more different server messages than client commands and - unlike the server - the client software is not forced to process every single message. Furthermore, the plug-in architecture of the clients is less restrictive than the modular structure of the server, and there are several messages that are relevant to more than only a single service (e.g., the registration of a new user). Also it should be possible to generate messages in a service module that are intended for the client's core system or other service plug-ins. Because of that, a direct addressing is not possible; the sID is only a hint to speed up internal processing (see chapter 5.3.5 on page 75).

To be able to keep track of all messages, numbers alone would not be sufficient, thus a human readable form was chosen. Additionally, the server offers a method to get a more detailed explanation for a certain code automatically, provided that the programmer of the service the code belongs to has maintained a special

code list. This is especially useful for error messages since all clients have a default error handling that will be used if the error was not processed by a service, generating a message like "Error in service Feedback: The category that was to be deleted does not exist" when the error message "FB!DCATNE" is received.

- The *message* has to be empty or must contain a single string if the message is an error message. This string is supposed to contain further information about the error (e.g., if a login failed, this string could be used to tell the user the reason for that). Otherwise, for normal messages, there is no difference to the command packet described above.

One problem with TCP is that there are no methods provided in the protocol to ensure that a connection is still alive, or to discover a connection loss in case the client or server have not been able to send an "abort" or "close" call [Pos81]. Since Java does not allow low level access to most computer or operating system features, a simple but effective keep-alive procedure is specified by the UCEP protocol: Beginning with the handshake, both server and client send a small packet every 10 seconds to each other. Whenever such a "ping-pong" packet is received, an internal counter is set to three. Whenever a packet is sent, this counter is decremented by one. As soon as the counter reaches zero, the connection is considered broken. This way, server and client usually notice a communication problem in less than 30 seconds.

5.3.3 Multicast

As an optional extension to the protocol described above, multicast may be used to deliver large data chunks to many participants at once. If this option is enabled, the server tells the clients during the handshake phase the current multicast group address and the port on which the multicast transmission will take place. Only the server will send to that address, all other participants are listeners only.

There are three conditions that have to be fulfilled before a message is transmitted with multicast instead of unicast:

1. The size of the packet's payload has to exceed a predefined threshold. The default threshold is 5000 Byte.

2. The message must be addressed to more than a specified number of clients; the default threshold is 10.

3. The creator of the message must have set a multicast flag. With this flag, the programmer can decide which messages may be sent with multicast.

When a multicast transmission is selected, the data payload is stripped from the message packet and replaced by control information. This multicast control information contains at least the total size of the data and an identification number, which is the current server time in milliseconds.

The original data payload is split into chunks of a predefined size (usually 1000 Bytes). These chunks are then sent sequentially to the multicast group, along with the identification number, a serial chunk number, the total number of chunks and a checksum. The receiving clients match the chunks with the corresponding checksum and if a chunk is received correctly, it is stored in a cache and can be accessed with the two identifiers it is sent with.

As soon as all chunks have been sent to the group address, the server transmits the modified message packet to each client using the normal unicast connections. When the clients have received this multicast control packet, they try to recompile the message according to the information found in the message, using the data received on the multicast channel that is stored in the cache. In case of any or all chunks missing, the clients send a resend command to the server with a list of all missing parts. As a response, the server sends the remaining data; both packets are transmitted using the unicast connection between client and server. This way it is guaranteed that all clients receive the message completely and error free.

Multicast transmission is completely transparent; neither server modules nor client plug-ins are notified about the way the data is sent or received. Furthermore, both peers can only prohibit a multicast transmission, but they can not enforce it; the server core, being the only component with a complete overview of the scenario, is the only decisive instance for that matter.

Several large simulations with up to 44 “real” PocketPCs and IEEE 802.11b wireless LAN were realised to measure the performance of multicast compared to unicast for the purpose of dispatching large data blocks to all clients. For the simulations a special testing module was written that sends a message of variable size (between 5,000 and 100,000 bytes) every minute to all clients, either as unicast or as multicast. The two parameters (size and delivery method) were determined by a semi random algorithm to eliminate possible side effects. As soon as a client had received a message completely and without errors, a receipt was returned - a very typical communication paradigm in WIL/MA. In each simulation, more than 2000 blocks were transmitted to get a statistically valid sample.

The results of two simulations are presented in Figure 5.4 on the facing page, one with 44 (shown in the upper half) and the other one with 22 clients. On the left side, the time measurements are charted. For each block size (x-axis) in kilobytes the average maximum time span (i.e., the longest time between sending the message and receiving the receipt) and the average minimum time span (i.e., the time between the transmission and the first receipt) are calculated, as well as the overall mean round-trip time for all clients. With 22 clients, the time needed to transmit a block to all clients is generally less than half as long as it is for unicast. Furthermore, with the double number of clients the mean round-trip time for multicast does not change noticeable (about 0.2 seconds with 100 kByte blocks), while for unicast it is almost doubled (from 4.5 to 8.6 seconds with 100 kByte blocks).

On the right side, the maximum, minimum and average number of bytes transmitted over the network is plotted for each of the two simulations. Only data bytes sent by client and server (including issued resend requests) are accounted for, not the size of network protocol headers or duplicated data packets caused by congestion in lower network layers. Both graphs clearly demonstrate that multicast causes much less network load than unicast. The number of chunks that have to be repeated is usually less than 4%. With an increasing number of clients, though, the chance of large resend requests rises; this can be seen in the frequency of random peaks in the red lines (few for 22 clients, frequent for 44 clients).

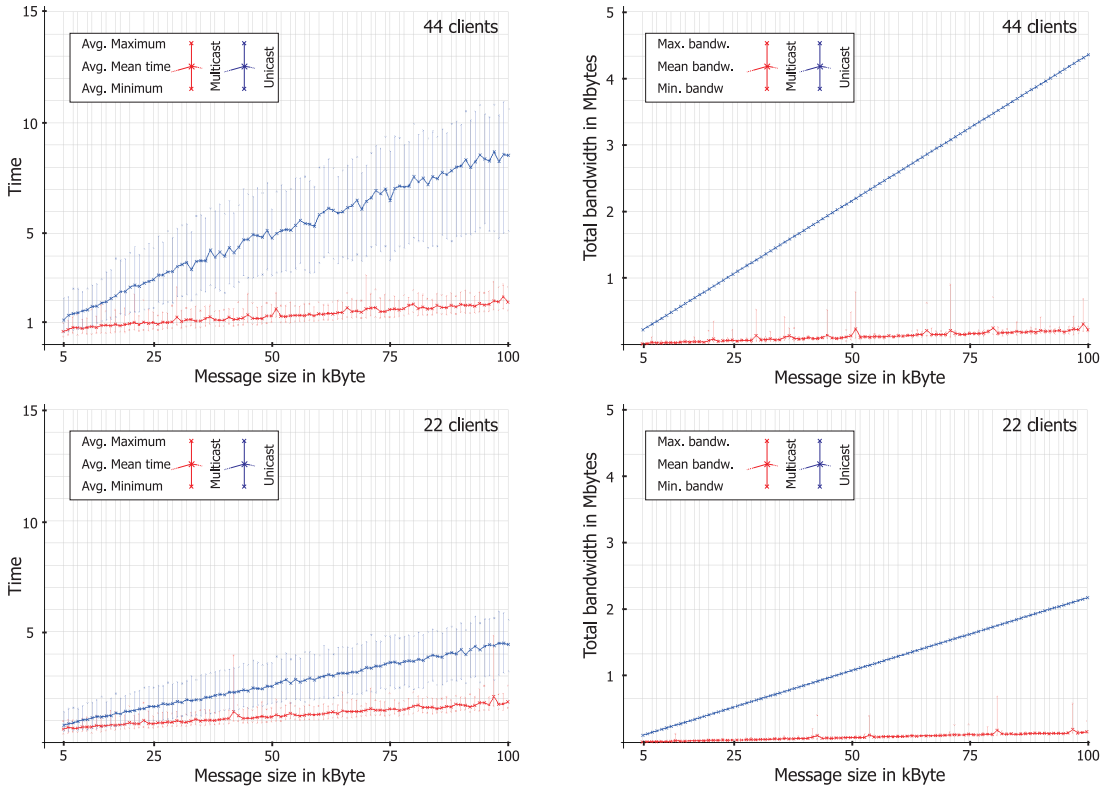

Figure 5.4: Multicast performace measurements with 44 and 22 clients

5.3.4 Messaging inside the Server

To enforce the integrity of the protocol, the server core provides a very strict messaging system that has to be used by all components of the server skeleton and the modules. It contains both a message creation and a message delivery component. For all modules and server components except the messaging manager, there is no way to access the outgoing communication channel directly.

For the messaging procedure, the `MsgBlock` class is provided. To create a new message, the static method `createMessage(WAM_Module module, String msgCode)` has to be called with two parameters: the calling module object (or `null` if it is a server core message) and the message code as described in chapter 5.3.2 on page 68. The new message object can later be filled with arbitrary information, using a number of methods for adding various data types (elementary types, strings or binary data blocks). When all data objects have been added, the `broadcast(String recipient)` method is called to finalise and send the message. All methods of the `MsgBlock` object

can be chained to make message creation as easy as possible:

```
MsgBlock.createMessage(this,"QZ-ANSSTR")
        .addInt(questionNum).addString(storedInfo)
        .broadcast(user);
```

The `broadcast` method delivers the current message object to the messaging manager of the server core. As a first step, the manager creates a clone of the message and encrypts and/or signs its contents if this was specified by the user (encryption will be covered in chapter 5.3.6 on page 78). Then a list of recipients is build, based on a number of parameters:

1. The mandatory recipient parameter given to the `broadcast` method is either a single name (only one user is supposed to receive the message), a single name preceded by an exclamation mark (all users *except* the user with the given name are potential recipients[4] or empty (no filter; all users connected to the server are put in the recipient list).

2. If a service module was specified as the origin, only recipients registered to that service will receive the message.

3. Optionally, a range of access levels can be specified. In this case, only users within that range are considered as valid recipients. This is generally used to send detailed update information to all teacher or administrator clients, but not to the students or vice versa. To determine the recipients, the service access level is used if a service is the origin of the message; otherwise the range is matched with the general access level.

The original message object is not altered and can be used to extend or reuse the information stored in it. This way, the code can be structured like this to ensure efficiency and reliability:

[4] This is used frequently to update all clients when data has been changed by the action of one specific student. Since that student automatically receives an answer (including the update information) from the server responding to his or her command, the general update message will be sent to all other students only.

```
MsgBlock.createMessage(this,"FB-UPDATE")
        .addInt(oldVal).addInt(newVal)
        .broadcast(null,User.ACCESS_STUDENT,
                        User.ACCESS_STUDENT)
        .addString(userName)
        .broadcast(null,User.ACCESS_TEACHER);
```

The message is then transferred to the message queues of the recipients which are responsible to deliver the contents of the message in the correct format as specified by the protocol. Message queues are individual and independent threads assigned to each user when he or she connects. Each queue has a FIFO list of outgoing messages and tries to send these messages to the corresponding client as soon as possible before entering an idle state. As soon as new messages are added to the list, the queue leaves the idle state and begins to send again. The independence of the message delivery ensures that even severe problems with one or more connections will not affect the stability of the overall system or degrade the communication with all other clients.

5.3.5 Messaging inside the Client

The client core classes provide a similar method for creating command messages: the `ClientCommand` class. Like the `MsgBlock` class it can be used to assemble arbitrary data to be sent as a command to the server. In contrast to the server, however, the client has to manage only one connection, thus no sophisticated message distribution system is needed. Instead, three simple methods are provided that can be used to deliver the command.

The difference between these three methods is the way the obligatory response messages from the server are handled. If the answer from the server is not relevant for further activity, the standard method is used:

```
int sendCmd(int sID, int cmd, ClientCommand data)
```

The method sends a command package with the given service ID, command ID and data to the server. The response packet will not be handled in any special way and, above all, the client will not wait for the server to process the command. This is the

default way to communicate with the server since many commands do not require an immediate feedback (e.g., acknowledging an update message from the server).

Other procedures do require feedback, but it doesn't have to be immediate. For example, if the client wants to poll the latest response to a question asked by the student from the server, it sends the command but continues its operations and displays the answer as soon as the response packet from the server has arrived. Command and response can be matched, if needed, by comparing the msgID field contained in the response with the return code of the `sendCmd` method. Although this can also be accomplished with the default `sendCmd` method, a convenience method was introduced to ease the implementation of new services:

```
int sendCmdActOnResponse(int sID, int c, ClientCommand data,
                         ClientActivity act)
```

This method also returns immediately after sending the command, but stores a `ClientActivity` object into a list against which every incoming packet is matched. As soon as a response packet with a matching message ID is received, this object, that contains an executable `run` method, is automatically activated[5]. All actions that have to take place after the reception of the message can be programmed into that method.

There are, however, occasions where the client has to be sure that a command was processed successfully. Examples are the registration to a service or sending the results of a quiz to the server. In these cases, a third variation of the send method can be used:

```
ClientEvent sendCmdAndWait(int sID, int cmd,
                           ClientCommand data)
```

Unlike the two methods mentioned earlier, this call does not return immediately with only the message ID. Instead, the client remains inactive and waits for the response to arrive. As soon as this happens, the response is parsed and returned to the calling method as `ClientEvent` object. The advantage of this method is that the results of

[5] The `ClientActivity` class is basically a small extension to the `Runnable` interface, providing some additional client-related functionality like access to the data inside the server response packet.

the command can be analysed in the same code segment that sends the command; this way it is much easier to handle errors and to prevent loss of critical data. But since it obviously has a negative impact on the responsiveness of the client's user interface, it should be avoided whenever possible.

The `ClientEvent` object - the return value of the `sendCmdAndWait` method - is an object that is generated automatically whenever a message is received, and it contains all data of the message packet. The name is derived from the Event/Listener-concept commonly used in Java to forward "events" to any number of "listeners". In this case, the event is an incoming message and the listener is usually the message handler of the client's main application. Some messages are intercepted by the client core; particularly the responses to commands sent by means of the `sendCmdActOnResponse` and the `sendCmdAndWait` methods. Whenever any other message is received (these may be responses to commands sent normally or update messages), the corresponding `ClientEvent` is forwarded to the message handler instead.

In a fixed order the message handler contacts each component of the client: first the main application (responsible for login, user management, etc.) then the various registered service plug-ins in ascending order of their IDs. If the message has a certain service specified in its service ID field (i.e., the sID contains a positive number), the corresponding plug-in is called first instead. By analysing the message code, each called component checks if it provides a handler for this type of message.

A `ClientEvent` object has a flag that is initially set to *"unhandled"*. Whenever one of the components has a handler implemented for the message, it may set the flag to *"processed"* or *"devaluated"*. If a message is flagged as devaluated, all further processing will be stopped and the message is regarded as dispatched. This is the most efficient way to handle messages that are by definition only intended for a single service. Messages flagged as processed, however, are still forwarded to the remaining components to give them the opportunity to evaluate the contents, too.

A default handler takes care of messages that have not been processed by any component. Normal messages are discarded and their message code is logged. If the message is an error message, though, a pop-up window will be displayed with the name and (if available) a description of the error.

5.3.6 Cryptography

During the implementation and evaluation of the WIL/MA software, the idea to use the input of the students for some kind of assessment - either as a part of the final grade or even as a replacement of an exam - came up very often. Unfortunately, the protocols used in the architecture (IPv4, TCP, UDP) provide no support for security. To make things worse, wireless LAN is known to be one of the most insecure methods to transmit data [BGW01]. The featured encryption protocol - called WEP - is very weak and has been successfully hacked some time ago. Furthermore, all participants in the same access point range also share the same keys, so WEP is only a protection against outsiders, but within the network there is no security at all.

This leaves the communication between server and clients open for any possible attack. Considering the quiz service as a possibility to rate the students, it would be easy for clever participants to sabotage the transmission of the questions, to disturb the flawless course of the exam and to intercept the answers of fellow students to copy or even change them.

As a solution to this problem, the WIL/MA software features a very strong encryption. Each message sent between server and client can be optionally and transparently encrypted and/or signed. This is simply done by setting flags in the appropriate transmit method. The receiving core system (client or server) automatically decrypts and verifies the message and forwards these flags in case of success to the recipient (client plug-in or server module). If the decryption or verification did not succeed, an error is forwarded instead. The recipient can then test if a message has been properly received and take appropriate action if a message that should have been encrypted and/or signed is received without any security measures.

The encryption used in this architecture is basically asymmetric, i.e., each participant generates a pair of keys. One key, the public key, is sent to the dialog partner, the other one, called secret key, is kept safe. A message can be encrypted with either key and only be decrypted with the other one, so if a message is ciphered with the public key only the owner of the secret key can read the message later [Sch96]. The keys are generated when the software (client or server) is first started and then stored for future use. As an option, the server can command the clients to discard stored

keys and force them to recreate their key pair as a security measure for important events. Public keys are exchanged in the handshake protocol when a connection between client and server is established (see chapter 5.3.2 on page 65) and immediately tested using a dummy message sent by the server and returned by the client. This ensures that the implementation of the used cryptography algorithms is compatible as this is unfortunately not always the case, especially when different kinds of devices are used.

A problem with asymmetric encryption as opposed to symmetric encryption, where only one key is used, is that the process of ciphering a message takes up much more time, particularly if the message is very large. Furthermore, if the server has to transmit a message to multiple receivers (e.g., quiz questions), the message has to be encrypted individually with the key of each receiving client. This would delay the transmission and further communication excessively.

Although there are protocols to exchange a key for a symmetric encryption securely (i.e., Diffie-Hellman), this alone would not be a feasible alternative: Using only a single key in a multi-user environment would not solve most of the problems stated above, and using multiple keys - one for each connection - would still require the server to encrypt broadcast messages repeatedly.

Instead, both principles are applied: The process of encryption is divided into two steps, an approach very similar to the well known PGP protocol used for secure email transfer. For each message, the first step is to randomly generate a symmetric key and to apply it to the data. In any case, this is done only once, regardless of the number of receivers. In step two, the symmetric key used in step one is encrypted with the public key of the receiver and then prepended to the encoded message. The resulting package is sent to the addressee. This step, of course, is repeated for every recipient, but the amount of data that has to be repeatedly encoded is reduced to about 16 bytes, which is the usual size of the symmetric key.

Encryption ensures that a message can not be read by an attacker. It does not, however, ensure the validity of the contents, neither does it guarantee the identity of the source: the public key is - as the name suggests - public and can therefore be used by anyone. To return to the example with the quiz: A student could still

intercept a message containing the answers of a fellow student, and while not being able to read the contents, he or she could reuse the encrypted message and transmit it to the server with his or her own name as sender. This "attack" is prevented by the concept of the signature.

In order to sign a message of arbitrary length, it is first transcoded into a smaller message with a fixed size - usually about 20 bytes. The arithmetic method to do this is a hash function and the resulting data set is called hash or fingerprint. This fingerprint has four main characteristics: it is not reversible (you can not reconstruct the message from the fingerprint), it is not constructible (it is very hard to design a message in order to get a certain hash), it is distinct (the same message always results in the same hash) and it is unique (it is almost impossible to accidentally have equal hashes for two different messages). As a second step, this fingerprint is encrypted by the sender using its secret key and then attached to the original message before it is ciphered. A fingerprint encoded in this way is called a *signature*.

The receiver deciphers the package, removes the signature and calculates the fingerprint of the message it has just received. Then it decrypts the signature using the public key of the sender and compares the result with the fingerprint. If it matches, the receiver can be sure that the origin of the message is correct since only the sender is supposed to know its secret key.

WIL/MA only provides a transparent mechanism to encrypt and/or cipher messages and vice versa with as little effort as possible. The required methods (for encrypting, decrypting, hashing, key generation) are provided by an external package, the BouncyCastle cryptography library that is licensed as open source. The different algorithms and their parameters are therefore easily exchangeable; the default is RSA for asymmetric encryption with 1024 bit keys, IDEA for symmetric encryption with 128 bit keys and SHA1 for creating 160 bit fingerprints.

Encryption can also be used with multicast transmissions; in this case, the asymmetrically ciphered key is part of the multicast control information contained in the modified message packet, see chapter 5.3.3 on page 70.

5.4 User Interface Design

In an interactive lecture, several interaction paradigms which are usually not in a traditional scenario with many participants are transferred to a computer-based communication channel. The entry point for the participants is the user interface on his or her mobile device. Since natural communication patterns (e.g., raising a hand to vote) are replaced by artificial actions (e.g., pushing a button), the user interface that already plays a very important role in the user-friendliness of a software system becomes an even more crucial part.

Unfortunately, several restrictions on the part of the mobile device complicate the development of the student client's user interface considerably:

- The screen size of PocketPCs is very small (about 5 inch in diameter); also the very low resolution (240x320 pixels) significantly restricts the number of elements (text, pictures, buttons) that can be displayed simultaneously.
- While point-and-click actions on a PocketPC with a stylus are closer to the natural pendant (using a finger to point somewhere) than using a mouse on a PC, entering text without a keyboard is a very slow and tedious process. When designing a user interface on a PDA, excessive text entry should therefore be avoided.
- The computing power of the PDA and the featured graphics processor is much lower compared to standard desktop PCs or notebooks. Therefore several user interface elements with excessive animation or other graphical effects should also be avoided.

To make things worse, the Java version that has to be used on PocketPCs does not support Swing, an advanced and in-depth configurable class package for user interface design that was introduced in Java 2. Instead, the outdated AWT package has to be used that only features heavyweight[6] GUI components. Heavyweight components have a fixed layout that tends to vary between the different platforms and usually

[6]Heavyweight classes feature direct calls to the corresponding methods provided by the operating system whereas lightweight classes are written entirely in Java

takes up too much space. Hence, it is not possible to use AWT components to create a user interface with a presentable and consistent design that is also easy to use and intuitive.

An extension was implemented, called DirectAWT, that solves at least some of the problems. It is a versatile and easily extendable class package with a three-level architecture.

The *DPanel* is the top level component. It is the only part of the DirectAWT package that uses and extends AWT to create a graphical surface; as such it can be used and placed just like any other AWT component. Only one DPanel object is usually required to build a reasonable user interface suitable for a PocketPC.

Being a crossover between AWT and DirectAWT, the DPanel is used to hold one or more *DPage* objects. Exactly one DPage is always displayed, switching between these objects is very easy and fast. A DPage contains all the information and interaction patterns available for the user in a specific context. The width of a DPage is exactly the same as that of the DPanel (usually full screen on a PocketPC), but the height is not limited. To access hidden portions of the DPage, the user can either use a traditional scroll bar or slide the page by pointing into an empty part of the page and moving the stylus up and down on the screen (respectively click and hold the mouse button while moving the mouse).

A DPage is filled with a number of *DItem* objects, the lowest object category in the DirectAWT hierarchy. DItems can contain any kind of information and define active regions that can either trigger an action in another GUI component or execute a process provided by the DItem. A DItem always uses the total width of the DPage but can be of arbitrary height. The accumulated heights of all DItems add up to the total height of the DPage. The WIL/MA software provides a small number of predefined DItems that can be used to assemble standard interfaces very quickly:

- Fixed components: Vertical placeholders and horizontal lines to visually separate different parts on a page. These components provide no interaction.
- Semi-fixed components: Text fields with optional active regions (similar to hyperlinks on web pages). For displaying text, a complex layout system is avail-

able which allows different fonts and font sizes, block text or centred text, text accentuation with bold, italic or underlined characters and full colour control.

- Checkboxes: Single checkboxes with a description or a row of checkboxes with numbers only. Both components can be assembled to button groups where only a single checkbox may be selected at a time (all others are unselected automatically). Single checkboxes may also be used as an active component, i.e., upon clicking into the checkbox an action is triggered. This is a very convenient way to build menus.

- Buttons: One or more buttons in a row with a short description.

- Text input: Small, single-line text fields suited for single words or numbers or full-featured text editors with multiple lines and word wrapping.

- Pictures: Presentation of automatically scaled pictures which can be zoomed and panned if they do not fit on the screen in their original size. Optionally, the pictures can be active and define regions the user can click into to trigger some action.

DItem objects can be reused anytime, but they must not be used on several DPages simultaneously. All information contained within is drawn with basic graphical operations featured by the Java graphics classes or by using convenience methods offered by DirectAWT (e.g., text display). Several frame buffers provide a very quick response whenever something changes and has to be redrawn. Even on standard PocketPCs it is thus possible to include simple animations.

The advantage of DirectAWT is that it is very easy to implement user interfaces for new services or to add features to existing services. The user interfaces created this way are well aligned and able to adapt to platform-specific constraints easily. An exemplary user interface created with DirectAWT can be seen in Figure 5.5 on the next page along with its Java code.

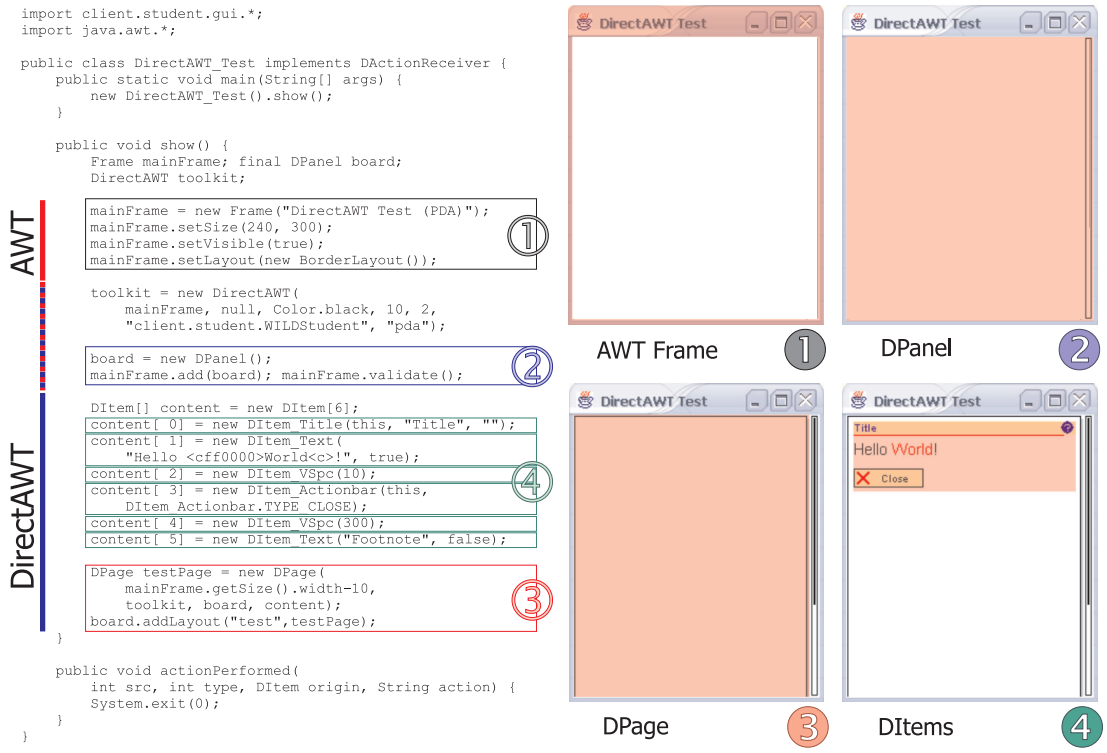

Figure 5.5: An example for creating GUIs with DirectAWT: Java code (left) and screenshots of the result (right)

5.5 Dispatching Tools

One of the biggest technical challenges was the configuration of 30 or more mobile devices to be used in an interactive lecture. The software had to be installed, network addresses assigned and default parameters set. Almost all PocketPCs provide a feature to backup and restore the system, thus enabling the user to clone a fully installed system to any number of devices of the same type. But still all devices have to be updated manually every time something in the setting or software changes.

It can not be expected of a teacher or an assistant to do these time consuming tasks all the time. Therefore, a number of tools were developed that help the user to dispatch an interactive lecture with as little effort as possible.

5.5.1 Dynamic Host Configuration

The Dynamic Host Configuration Protocol, in short DHCP, is used in LANs to assign IP addresses along with a number of network related data to attached computers automatically [Dro93]. Usually each computer, PocketPCs included, supports DHCP to receive a network address. WIL/MA provides a simplified version of a DHCP server to dynamically assign IP addresses to the mobile clients. It is an optional feature that can be used if no other means of automatic network configuration is available.

DHCP allows easy network configuration of a large number of devices simultaneously without the need of manual interaction. Addresses are assigned from a preconfigured pool on a first-come-first-serve basis; the lease time is 48 hours. Once an address is assigned, it is reserved for a particular network adapter until the lease expires, even when the server is restarted during that time. Thus, it is impossible that two devices are assigned the same IP address. If desired, a list of MAC addresses can be provided to assign fixed IP addresses to a number of devices.

5.5.2 Quick Login

The user of a client has to provide at least four pieces of information to log in: user name, password, IP address of the server and the port to connect to. Unfortunately this was a steady source of failure, particularly when there was not enough time to explain the usage of the software to the participants properly.

The Quick Login feature helps to solve this problem. Whenever a client is started and the login dialog appears, it sends a request for configuration on a common multicast group address. The server answers on the same group with a response packet. A unique ID ensures that only the client the response packet is intended for uses the contained information.

In the most basic mode, only IP address and port of the server software are forwarded to the clients. When this information is received, it is automatically filled into the corresponding text input fields, which are then disabled. Two more advanced modes also create an anonymous user login reserved for the calling client and send the

username along. Depending on the configuration of the Quick Login feature, the user can still change the user name and password to a registered login or is forced to use the anonymous login. In this case, the login screen disappears automatically upon reception and the login procedure begins. This way, it is very easy to use WIL/MA in large conferences without having to register all participants ahead of time.

The clients use the first valid response packet received and discard all subsequent responses. In multi-server environments this means that the clients are distributed almost perfectly among the different servers with a preference to the one that has the smallest round trip time (i.e., the spatially closest server).

5.5.3 Automatic Updates

The WIL/MA software basically consists of several applications installed on a multitude of independent devices. These applications have to interact very frequently and therefore it is very crucial that all parts are perfectly aligned with each other. Whenever the software is updated or changed, this usually affects both the server and the client. This means, that with every change in the software all participating devices have to be updated. This is both time-consuming and error-prone.

As a solution, an automatic update feature was built into the WIL/MA system. The clients only need a tiny subset of fixed classes for the most basic operation, i.e., logging into the server and requesting all other required classes which are provided by - and thus perfectly compatible to - the server.

To be able to do this, the default class loader of the Java core class package was replaced. Instead of loading a required class from a file or archive, the class loader requests a binary representation of that class from the server and stores it in a cache. Subsequent requests for the same class (even after a restart of the client software) are then satisfied locally to save time and bandwidth. With each class stored in a cache, a hash number of that class is recorded. These hashes are matched against a list received by the server when the client is started; updated or otherwise changed classes are removed from the cache to be loaded anew when they are first needed.

5.6 Services

Until now, the WIL/MA software is a communication system, allowing easy and reliable data transfer between a server and a multitude of clients installed on detached mobile devices. Little of the functionality described so far, however, is visible to the typical user. The software experienced by the user is almost completely made up of service modules and corresponding plug-ins for the clients.

Each service describes a specific interaction paradigm between the students and the teacher. It offers methods to project the natural interaction onto computer-supported data transfer and to help the lecturer to handle all data received from the students. Services have two main attributes: the level of continuity and the level of attendance required by the participants:

- The *continuity level*: most services are offered all the time throughout an event and therefore have a high continuity level. Other services are only usable during short time periods, usually triggered by the lecturer.
- The *attendance level*: services with low attendance level only require simple input (like clicking a button or moving a slider), while other services depend on more sophisticated input (e.g., entering some text).

Other characteristics, like cognitive load for teacher and students or interoperability with other services, are either hard to determine or can be easily derived from the two introduced variables.

5.6.1 Online Quiz

Beginning with the Online Quiz, being the most intuitive and appealing of services, we will now describe in detail the three interactive services included in the WIL/MA software at the moment[7]. A quiz consist of one or more questions about the subjects that were presented during the last one or two lectures. The answers have to be automatically analysable in order to be able to discuss the results immediately after

[7] as of January 2005

the quiz. So usually multiple-choice questions are used where either exactly one answer out of four is correct or the students have to figure out for each answer individually whether it is correct or not. While scoring for simple multiple-choice questions is very easy (all or nothing), it is more complicated for real multiple-choice questions. Several models are offered by the software, ranging from very fair scoring (each correctly marked answer gives an equal share of the total score) to very hard scoring (wrong answers remove points, so the total score may be negative) [FJM98].

But there are other types of questions, too. The cloze, as the first advanced quiz type, is a text where a word is missing, or a question with one word as the answer. The students have to enter this word on their device and send it to the server. The teacher can provide a list of correct terms the answers are matched against; it is even possible to value these words, e.g., "IPv4" would give full score while "IP" would only result in 75% of the score when the question is "What layer 3 protocol is generally used in the Internet today?" He or she can also determine the level of generosity towards spelling errors. If abbreviations are asked for, the answer has to be error-free and case-sensitive. Complicated technical or scientific expressions, on the other hand, only have to be recognisable; one or two typing errors should not lead to a lesser score.

A derivate of this question type is the arithmetic problem where a numerical answer is expected. The teacher can determine the required number of decimal places and two levels of allowed variation. Answers within the interval of the first level of variation will get full score; answers within the interval of the second level will result in a reduced score.

A third type currently implemented is the Clickable Image. The students are sent an image along with the question(s) and have to mark a single point inside this image as the answer. This type of question can be used to ask questions like "Where on the given map is Sri Lanka?" or "Mark the location of the error in the given graph". At the moment, only one region can be specified as correct, the students get either full score or no points at all.

All answers are analysed and accumulated by the server module. The global results are then displayed by the administrator client to be projected for the class. Each

question is shown individually with the question text, the correct and wrong answers and a graph showing the results of the students. The teacher is then supposed to discuss the question and explain why e.g. an answer marked by many students as correct is actually wrong.

Furthermore, each student gets a very detailed feedback sent to his or her computer. In addition to the information displayed with the projector, the student is also shown his or her own answers in comparison with other students, the total score of the quiz and the performance compared to the rest of the class and to the previous quiz. For the latter, an internal weighted score is calculated that may also be used by the teacher for an advanced analysis. The formula to construct the weighted score of a single question is:

$$w = (\frac{s}{t})^{\log_{0,5}\frac{m}{t}}$$

with m = mean value of all students' scores, t = the maximum achievable score and s = the student's own score. The weighted score is always between 0 and 1. Furthermore, w is 0.5 if the student's score is exactly equal to the mean score, which is used in this formula to represent the difficulty of the question. For a complete quiz sheet, the weighted score is the mean value of all weighted scores of the questions contained in it.

Last but not least, the teacher can decide to publish ranking lists where the students can see their performance in comparison to their fellow students. The ranking score is the mean value of the student's score in all registered questions multiplied with the square root of the number of questions the student has participated in. These scores are included in the list, so the students can exactly identify their standing.

Obviously, this service is very interruptive and demands a high level of attention from the participating students. For each single question, the teacher has to reserve about one to two minutes, which are lost for the "normal" lecture. On the other hand, the lecturer and the students get a better understanding how well the subject was understood. Sample screen shots of the quiz service can be seen in Figure 5.6 on the following page.

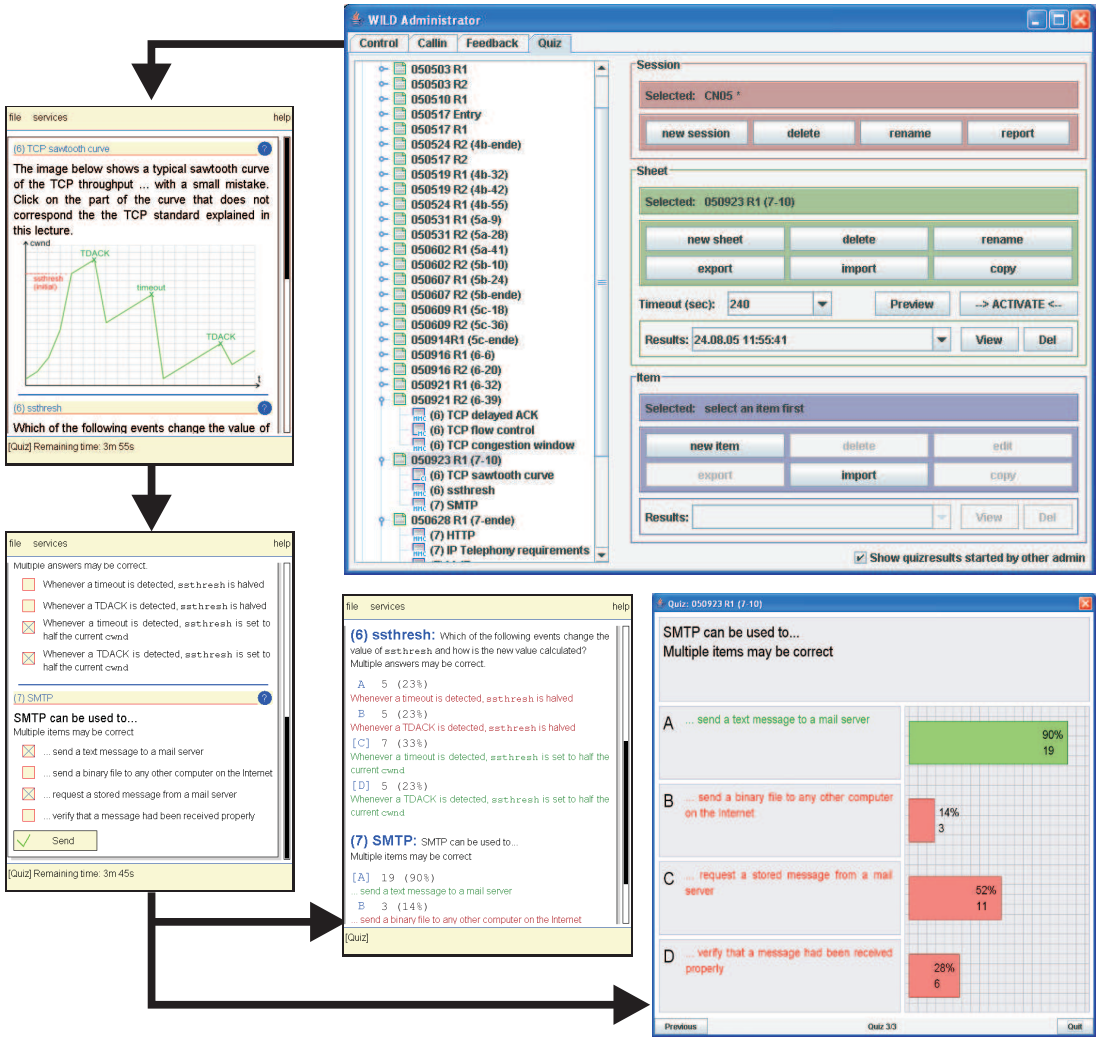

Figure 5.6: Screenshots of the quiz service of both student client and administrator software

5.6.2 Online Feedback

The second service included in the WIL/MA software is the Online Feedback. A very unobtrusive service that does only require very little attention, the Online Feedback gives the audience the possibility to give instant feedback at any time to one or more categories suggested by the lecturer. Categories could be "is the lecture too fast or too slow?", "are you missing previous knowledge for the current lecture?" or in remote scenarios: "is the audio and video quality sufficient?".

The students see all available feedback categories on their device as slider bars with different colours. Negative regions are displayed in red, positive regions are green and regions in-between have different hues of yellow. By clicking into the slider, the selected region will be marked with a slider symbol and the feedback given in this way is forwarded to the server. Categories can either range from negative to positive values (the level of approval of a given statement is measured), from positive to negative values (similar to the German school grade system) or from negative to positive to negative regions (e.g., "too slow" - "just right" - "too fast"). For ease of handling, the slider is divided into an arbitrary number of fragments; usually five or seven areas are used. Optionally, the feedback categories may offer the students to abstain, which is then the default when a category is started.

The administrator is shown a bar graph displaying the current distribution of feedback and a gradient graph displaying the progression of the mean values of the feedback over a selectable time span. The students usually do not see this information, although it is possible to "open" a category. In this case, the students, too, will see the bar graph on their device. Two screen shots of the online feedback service in Figure 5.7 on the next page show how student and teacher usually perceive this service.

Especially for psychologists this is a very interesting tool. It allows to measure the course of different aspects regarding motivation or attention in learning scenarios in a very convenient way. Furthermore, the data is collected continuously *during* the lecture, and has not to be reconstructed afterwards.

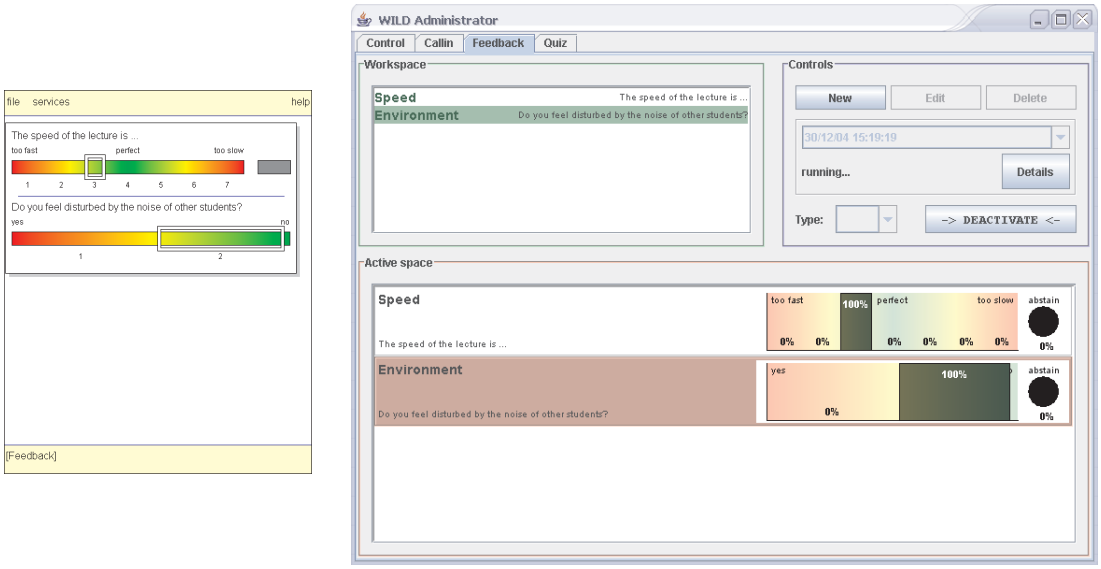

Figure 5.7: Screenshots of the online feedback service of both student client and administrator software

Depending on how important the teacher regards the feedback input, he or she can choose to remind the participants after several minutes of inactivity automatically with a short message on their display. Furthermore, to keep the feedback up to date, it is possible to set a timer after which a feedback becomes invalid: the slider of the student will then return to the default position which is either the most positive value or abstention.

5.6.3 Call-In

Most people (usually for no reason) are afraid to disgrace themselves in front of other students, i.e., by asking stupid questions or giving a blatantly wrong answer to a question. A solution to this problem is a service we call Call-In. Its predecessor Virtual Handraising only allows students to set a flag which means that they have a question or remark and want to be called soon. The Call-In service, though, also enables the student to send a complete message to the lecturer containing the question or remark. This is per default anonymous - only the software system knows where to return the answer.

All messages are collected, and when the lecturer has some time - e.g., during a quiz

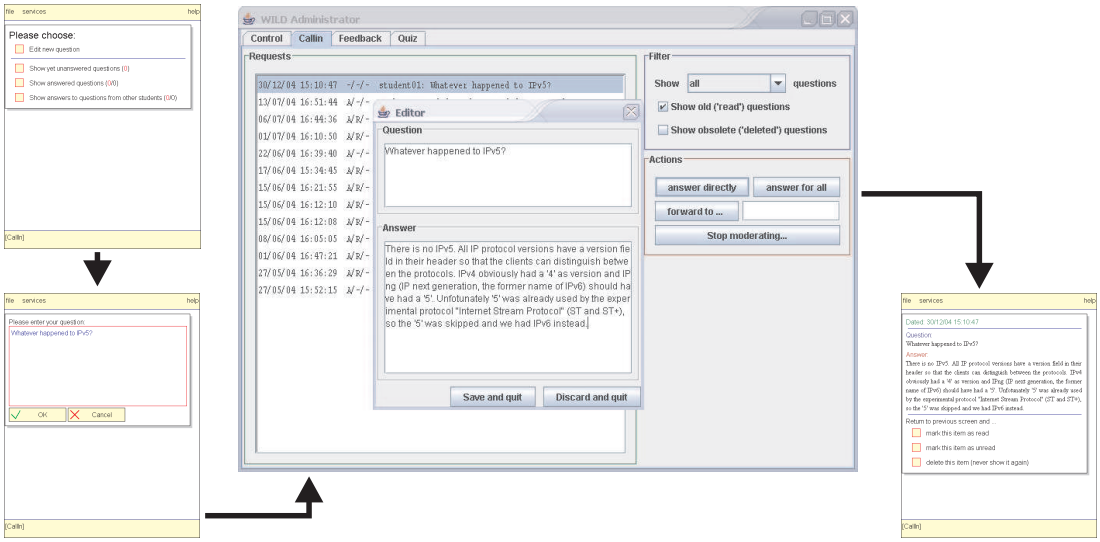

Figure 5.8: Screenshots of the call-In service of both student client and administrator software

round - he or she can go through the list and respond accordingly. While answers to simple questions can be sent directly to the questioner, more interesting questions should be answered publicly.

The Call-In service doesn't interrupt the lecture because all messages are stored for later review, but it requires some attention by both the student and the lecturer. The student has to formulate and enter the message while the lecture goes on, so he or she may have some problems trying to catch up on the missed parts. The lecturer, being obviously unable to process the list during the lecture, will also have difficulties to respond to all incoming messages in the few short breaks, particularly when many students are participating. He or she may thus require an assistant to be able to offer this service properly.

The interface for the lecturer or the assistant shows a list of all incoming questions. A question can be viewed at any time by clicking on an entry. In the following dialog, the administrator can choose to enter the answer to the question directly and then send it back to the student or to all students (of course, the identity of the questions' origin remains hidden in this case). Alternatively, he or she can forward the question to the teacher on whose screen the question will appear as a post-it-like notice.

The students' user interface is build like a mail client. They can write new questions and access lists of questions not yet answered, answers to their own questions or answers to questions asked by other students which have been answered publicly. Answers can be marked as "read" to be able to distinguish new answers, or be deleted. In either case, a textual representation of each answer is copied to the student's device as a file that can be copied to another computer later. Some screenshots from the students' client and the administrator software can be seen in Figure 5.8 on the preceding page.

According to the design principles of services described in chapter 4.4 on page 53, all information is stored solely on the administrator's computer. This way, each student can always access all answers or question stored for him or her, regardless of the computer he or she currently uses.

6 Evaluation

Beginning with detailed descriptions of each experiment, the chapter will present some technical and conceptual experiences including a small survey of currently available devices and their advantages and disadvantages. After that, as the major part of this chapter, all important results of our field studies will be discussed. Two experiments with WIL/MA performed at Stanford University will conclude this chapter.

6.1 Experiments and Field Studies

The development of the tools was accompanied by thorough testing and experimentation. Beginning in early 2002, we designed and conducted a total of six experiments and field studies in very close cooperation with the department of educational science at out university. Our aim was not only to prove - or disprove - the positive effects the interactive lecture is supposed to have on the students (higher motivation, better learning success), but we also included detailed questionnaires to learn about flaws in our software to be able to correct them and to get ideas how to improve the scenario generally. Furthermore, we investigated certain aspects of the setting in order to create useful guidelines for teachers who want to use the WIL/MA tools most effectively [WS05].

All our experiments and field studies took place at the University of Mannheim in courses of two different faculties. Their realisation was financed by the BMBF (Federal ministry for education and research) as part of the "VirOR" project (virtual university of the upper rhine valley, [Obe05]), by the Learning Lab Lower Saxony (L3S, [Sax05]) and particularly by the DFG (German science foundation) with a

	Winter 2001/02	Summer 2002	Summer 2003	Winter 2003/04	Summer 2004
Course	Multimedia systems	Computer Networks	Computer Networks	Paedagoical Psychology	Computer Networks
Objectives	Technical and didactical trial	Comparison traditional vs. interactive	Teacher feedback variation for quizzes	Individual vs. social benchmark in quizzes	Observation of a "normal" IL
Indep. variable	Interactive lecture vs. Traditional lecture with same topic	Lecture with a traditional and interactive phase	Systematic variation of feedback for quizzes	Systematic variation of the feedback benchmark	–
Dep. variables	Acceptance and learning success (pre-post)	Acceptance, self-efficacy, learning success (pre-post-follow up)	Acceptance, learning success (pre-post-follow up)	Acceptance, learning success (pre-post-follow up)	Acceptance, learning success (pre-post)
Sample	N = 44 (9 ♀, 35 ♂)	N = 99 (9 ♀, 92 ♂)	N = 54 (6 ♀, 48 ♂)	N = 69 (57 ♀, 12 ♂)	N = 70 (3 ♀, 66 ♂)
Devices	14 Notebooks 3 PDAs	27 Notebooks 3 PDAs	∼ 20 Noteb.[1)] 20 PDAs	70 PDAs	∼ 12 Noteb.[1)] 45 PDAs
Software	UCE-Tools	UCE-Tools	WIL/MA	WIL/MA	WIL/MA

Notes:

[1)] The students could bring their own notebooks to the lecture

Table 6.1: Summary of experiments at the University of Mannheim

grant for the project "Lehr-Lernexperimente zur Wirkung von Feedback in WaveLAN-unterstützten interaktiven Vorlesungen" (DFG project no. 649/18-1, [LL05]).

As the sixth experiment was still work in progress when this dissertation was written[1], only the first five studies will be discussed. A summary of these is given in Table 6.1.

January 2002: First Try-out

In a first try out in the Winter semester 2001/2002, we investigated the technical operability of a very early version of the WIL/MA tools. Also, we wanted to get the

[1] as of January 2005

first hints about the usefulness of the interactive setting and as many suggestions for improvement as possible [SME+02].

Therefore, two lessons of the course "multimedia technology" (senior course at the department of computer science) were selected: "operating system support" and "content analysis of still images". Lacking a large pool of mobile devices, we divided the students into two groups and asked the professor to read each of the two lessons twice (once for each group of students). In two of these altogether four experimental lectures, we prepared 17 notebooks and 3 PocketPCs for the students. These devices had a client installed that allowed the students to participate in three quizzes per lecture with three questions each; being a prototype, the early WIL/MA tools provided individual clients for notebooks and PocketPCs with a different behaviour. While group A had access to the interactive devices only during "operating system support", the students of group B could test their knowledge in the quiz only in their second lecture, "content analysis". This design made it possible to compare the two settings (traditional lecture vs. interactive lecture) within a group with two different lessons and between the two groups but the same lesson.

Before and after the try out, we asked the students to fill out a knowledge test of 24 questions regarding the topics of the two selected lessons. The difference between the test before the experiment and the test after the experiment was counted as knowledge gain. Furthermore, the students rated an extensive number of aspects regarding the tools, the lecture and the general setting in questionnaires after each lecture. To get a better idea what the students expect from an interactive lecture or what problems they had encountered in the try out, we invited them to a group discussion afterwards.

A picture of a group of students using PocketPCs and two screenshots of the tools that were used at that time can be seen in Figure 6.1 on the next page.

May-July 2002: Evaluation in Computer Sciences

In the following semester, we conducted a larger experiment in another senior computer science course, "computer networks". This time, our intention was to reconfirm

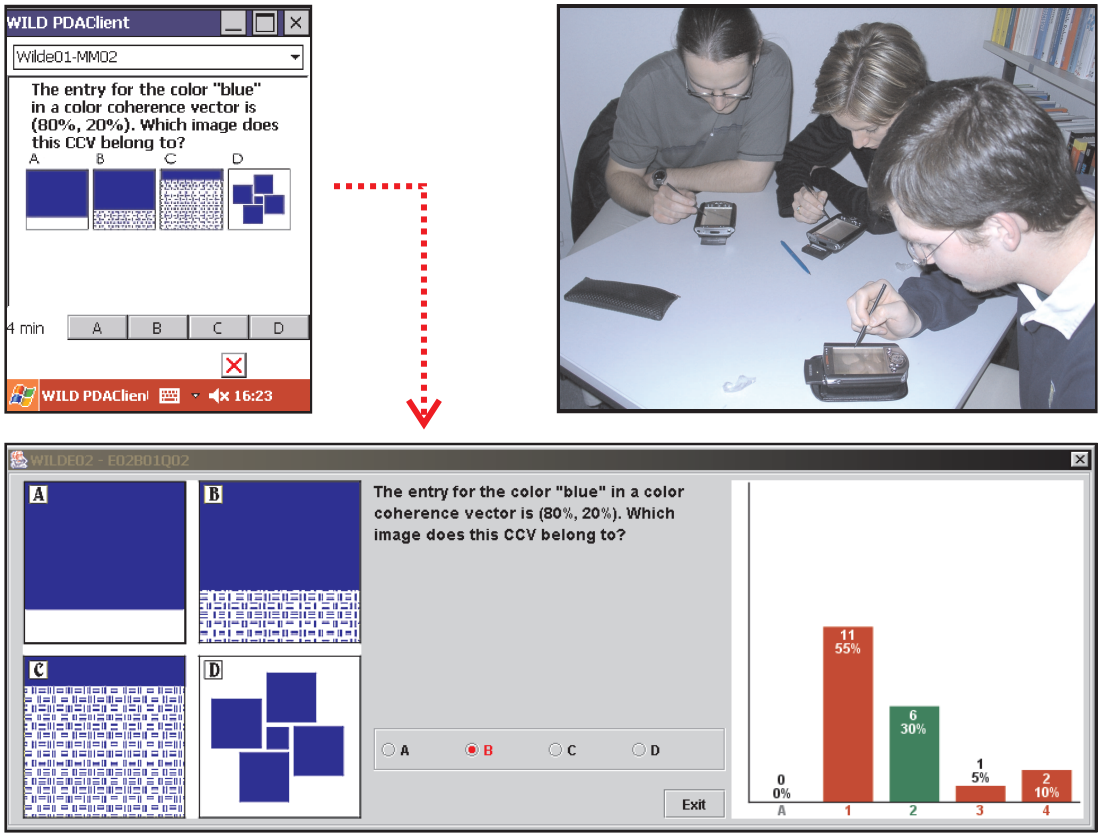

Figure 6.1: Screenshots of an early WIL/MA prototype used in the first try-out (top-left and bottom). Photo of several students using PocketPCs in the try-out (top-right).

several results of the try-out in a more realistic scenario and with a larger sample [SME+03].

The complete course was divided into three parts; parts one and three were not altered, part two - four consecutive weeks with two lectures per week - was improved with our WIL/MA tools. During the interactive phase, we equipped half of the approx. 70 local student for the first four lectures and the other half for the rest of the time with laptops. The lecture was a tele-cooperation with the University of Freiburg, so about a dozen additional students were attending the course from a remote classroom by means of video conferencing. These students had access to their own laptops in all interactively enhanced lectures. On all laptops, a newer version of the client software was installed that additionally included the call-in service. In Figure 6.2 on the facing page you can see a picture of the lecture hall with the laptops

and a screen where the students of Freiburg were displayed, Figure 6.3 on page 100 shows the experimental design.

Knowledge gain was measured again with a number of knowledge tests: one at the beginning of the course, two more immediately before and after the interactive phase and another one a few weeks after the exam. The knowledge tests comprised of 80 multiple choice questions. These questions were carefully selected out of a much larger pool, so that each of the 8 topics of the course was represented by the same number of items for each of the three phases of the learning cycle proposed by Mayes [MCTR94]. Additionally, the students had to fill out a small web-based questionnaire after each lecture and more detailed surveys at the end of each phase. After the interactive part, we also invited the students from Freiburg to a group discussion.

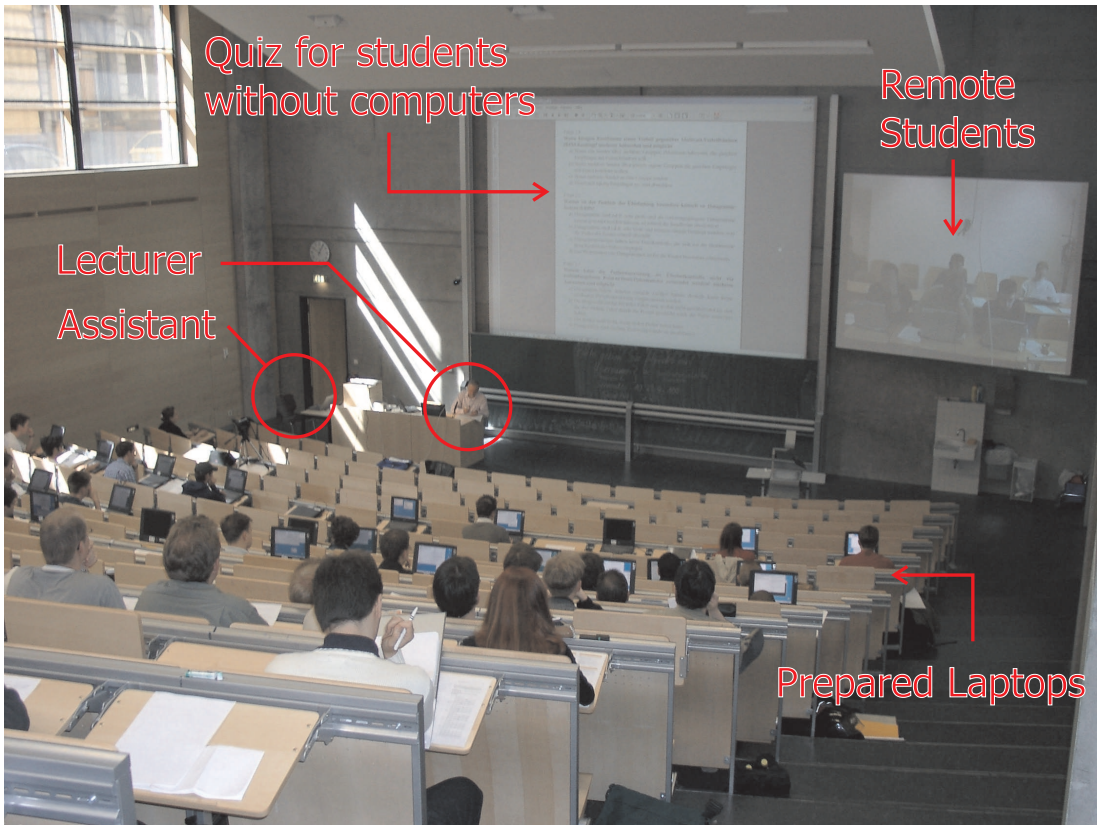

Figure 6.2: A photo of the 2^{nd}experiment in a computer networks course with remote students.

A rather annoying disturbance in this experiment was the soccer world cup in South Korea and Japan. Many of the matches were played at the same time the lectures

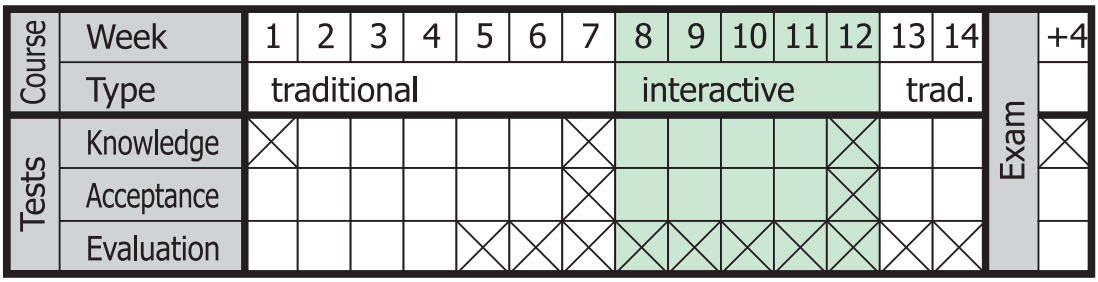

Figure 6.3: The design of the 2^{nd} evaluation

took place, so we worried that many students would rather watch an exciting soccer game than go to the university. Fortunately, our fears were mostly unfounded; enough participating students faithfully attended the lectures.

For the first time, we additionally quantified the course of the students' motivation, interest and strain throughout a lecture by asking the participants to fill out a short form at five equidistant points in time during each lecture.

May-July 2003: First Quiz Feedback Experiment

One year later, we used the same course for our third experiment. In this experiment, we replaced the old prototype with the first release of the new WIL/MA version that still lacked many of the features described in chapter 5 on page 57, but already had an unified client user interface for all the different platforms.

In this experiment, the WIL/MA tools were used in all but the first three introductory lectures. Using a pool of 22 new PocketPCs and encouraging the students to bring their own computers (usually notebooks) to the lecture, we were able to equip almost all of the 50 students with devices suitable for WIL/MA. Again, we concentrated on the quiz service with two quizzes of three questions each in each classroom session. Given that quizzes are a very time consuming service, it was our intention to find out how detailed the feedback of the teacher about the results of a quiz has to be: is it sufficient to only show the correct answers along with the accumulated answers, of the students or does the teacher have to explain why an answer is right or wrong?

Therefore, we asked the professor to not comment the results of the quizzes at all in

the first part. In the second part, only those statements of the multiple choice questions were discussed that were marked as correct. A full discussion of the complete question took place in the third part of the course. In a short, fourth part, we skipped the discussion and public review of the results completely and sent this information to the students' devices directly instead.

The knowledge test of the prior experiment was reused. The students had to fill out the test before and after the course; relevant parts of the test were given to the students between the phases as well. A shortened version of the online questionnaire also was due after each lecture. The design of the study is shown in Figure 6.4.

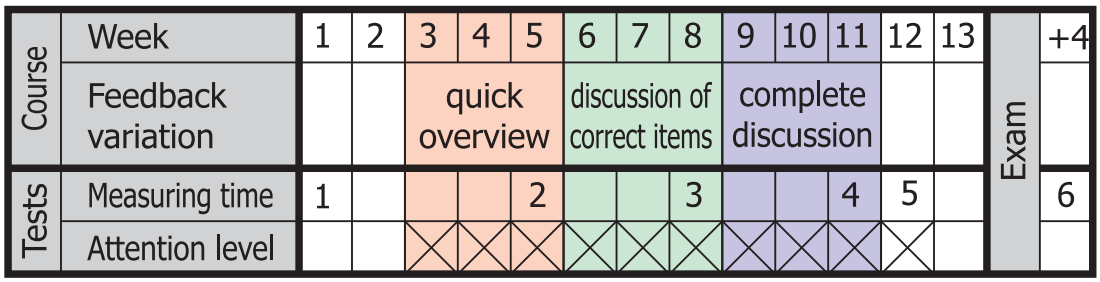

Figure 6.4: The design of the 3^{rd} evaluation

November-February 2003/2004: Second Quiz Feedback Experiment

The first three experiments took place in computer science classes where the students and teachers are usually very skilled in using computers or other electronic devices. Therefore we decided to implement the interactive tools in another course: "pedagogical psychology", a senior course at the department of psychology. A more complete set of WIL/MA tools was used this time, particularly the user friendliness and error handling had been improved. Also, we received a grant to purchase another pool of 50 new PocketPCs. Thus a total of 70 students could be equipped with WIL/MA devices [SWE04, SSEW04].

Surprisingly, the course was attended by more than 150 students, so despite the large PocketPC pool, we had to split the students into two groups. Students of the first group could pick up a PocketPC at the beginning of each lecture, students of the second group had to use pen and paper forms instead. The only service that was used

in this experiment was the quiz service, but this time, we deliberately removed the statistical analysis of the students' answers from the projector view. Only the teacher was able to see the detailed analysis, the students were sent none or only aggregated information which could be an individual benchmark (e.g., "you have done slightly better than in the last quiz") or a social comparison (e.g., "most of the students have a lower score, very few a higher score than you"). Therefore the students with PocketPCs were in turn divided into three groups and the twelve lectures into three phases. The three different feedback alternatives were then permutated over all phases and groups. A summary of the design is shown in Figure 6.6 on the facing page, some photos during and after a quiz are presented in Figure 6.5.

Figure 6.5: Photos of the 4^{th} field study

Again, extensive knowledge tests and questionnaires were used to measure acceptance and knowledge gain. Also the trend measurements were repeated; this time, though, the data was also collected with PocketPCs. Since many students did not have much experience with computers, we gave a 20 minute introduction for the PocketPC users at the beginning of the course.

May-July 2004: "Free-style" Field Study

All experiments so far involved a lot of surveys, carefully planned timetables and compensation money or other incentives for the students. While this granted us

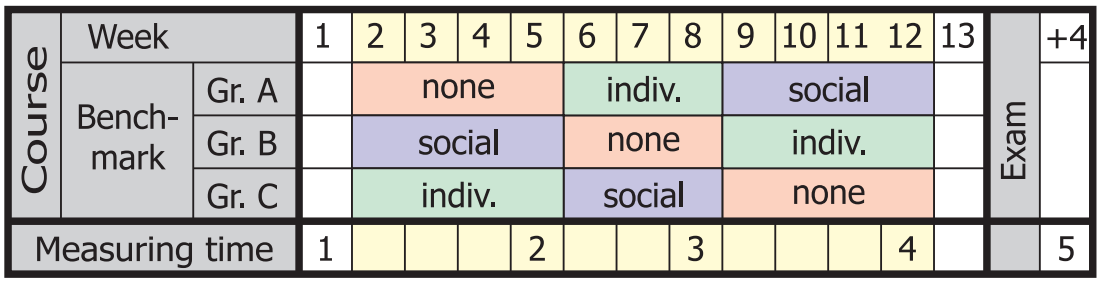

Figure 6.6: The design of the 4^{th} evaluation

extensive and reliable data, it still did not answer the question how the students would react to a normal day-to-day application of WIL/MA.

To investigate this was our intention for this fifth experiment. Participation was completely voluntary, and the students were given the impression that WIL/MA, though being a fairly new technology, was from that time on a standard part in the "computer networks" course. Many students brought their own notebook or PocketPC to the lectures, all others could borrow one PocketPC from our pool for the entire semester. About 30 to 50 students ($\approx$ 75% of all students in that course) actively used the WIL/MA software in each lecture. Knowledge tests and questionnaires were reduced to a minimum (a pre and post knowledge test combined with a short survey to measure the acceptance).

Instead, we used the chance to try out some parts of WIL/MA we had neglected before. All three services implemented so far were used: quiz, online feedback and call-in. The results of the quiz service, which - for the first time - did not consist of multiple-choice questions only, but also featured the new "clickable-image" question style, were discussed elaborately. Additionally, the students received further information to their devices (social and individual benchmarks) and a ranking list was updated and published after each quiz.

To compensate the possible effects of yet another major sport event (this time the European soccer championships), we offered a small competition with the idea that the students would team up as one of the competing countries and play against each other according to the game plan of the championship. Unfortunately, only a dozen students were interested to participate; probably an effect of the early drop-out of

the German national soccer team. Therefore a smaller match between only three teams was set up to try out group functionality as one of the new enhancements of the software toolkit (see chapter 7.2 on page 130).

6.2 Technical and Conceptual Results

Although the experiments were primarily designed to give answers to didactical questions, we were able to learn much about the technical feasibility of the setting, the conceptual design to set up all required electronic devices and to supply the students with PocketPCs or notebooks in the short breaks between the lectures. Beyond that, we tested software and hardware on a regular basis to detect serious bugs before using the system in a classroom or simply to determine the limits of the technical equipment.

6.2.1 Stress Tests

Three different system tests were implemented and used frequently:

- A *software stress test* was originally implemented to find out if Java is at all suited to implement our kind of scenario. Using a standard cable-based network, multiple automated clients running on several different computers (about 8-10 clients per computer) connect to a central WIL/MA server. These clients then send commands to a specific server module with a very high frequency and analyse the answers that are automatically returned. CPU usage of the server, average round-trip time and network load are measured. Errors, like missing answers or wrong answers, are written to a log file.

- The *hardware stress test* has a very similar concept, but instead of many simulated clients from hard wired computers, the "real" mobile devices and a wireless LAN is used. When all devices are connected, the server starts to send very large packets to each client which are promptly returned. The frequency and the size of the packets or the number of served clients is low at the beginning

and increased steadily until the network becomes unstable and the round-trip times exceed a certain value (usually 1 second).

- *Data validity tests*, last but not least, are a complete simulation of an interactive lecture. Client bots - both for the student and the administrator client - randomly send valid commands to the server and react accordingly. For example, the administrator bot fires a quiz round at a random time and the student bots send random answers. Additionally, problems like connection losses or "sleeping students" are simulated to see if the software system can handle these situations properly. A special observer client regularly checks if the data is still valid and fully synchronised between the server and the clients.

The server was never a limiting factor, as long as it was installed on a decent notebook or desktop PC with an operating system with good network support like Linux or a professional version of Microsoft Windows. The most recent test featured 360 simultaneous connections from 10 different machines with two 1000 byte packets per second per connection (one from the client to the server, one from the server to the client). After three hours, the CPU load of the server computer - a 1500 MHz Intel Pentium IV with 512 MB RAM and a 533 MHz front side bus - was rather high indeed, but the round-trip time was consistently noncritical (<600 ms).

A limiting factor, however, is the wireless network, at least when 802.11b is used. Using the average data rates from the 4^{th} experiment, the latest hardware stress test resulted in a statistical limit of 145 concurrent clients if the server is connected to the access point with an Ethernet cable. If the server computer also uses wireless LAN, a maximum of about 60 clients should not be exceeded. It can be assumed though that with 802.11g these limits will rise considerably.

Regarding the stability of the software system, our experiences in the studies showed that WIL/MA operated flawlessly with only very few exceptions.

6.2.2 Choosing the Right Device

In the field studies, we used a wide range of possible devices: PocketPCs and notebooks with different operating systems, computers we had installed and configured

and computers owned by the students. Many initial problems, sometimes critical, had to be solved before the tools could be used smoothly on all these different pieces of equipment. However, given the chance to work with such a huge diversity, we learned much about all the individual computer platforms currently available. As the most interesting devices for interactive classroom scenarios we identified the PocketPC described in 3.1.1 on page 29, notebooks and TabletPCs, which are a crossbreed between a PDA and a notebook.

As far as CPU power and memory are concerned, these were not limiting factors in most of our experiments and thus no criteria for selection. In other words, any modern device has enough computing power for the typical interactive classroom use. There may be special applications beyond the interactive lecture, though, where the CPU power of the current PDAs may not suffice; these include real-time video playback and complex graphical simulations (e.g., 3D graphics). Similarly, the storage capacity of all devices was always sufficient for our scenario. The PDA has no physical hard disk but the PocketPC operating system simulates a virtual disk drive in main memory. If more memory is needed than the PDA provides, most devices are easily upgradeable up to 2 GBytes.

Battery lifetime was a major surprise in our field studies in Mannheim: We expected PDAs to have a much longer lifetime than notebooks. This was not the case; a high-end PocketPC with colour display and WLAN enabled does not run much longer than 90 minutes. The older notebook PCs had the same problem, so there is no clear advantage of either device. The consequences are that all devices have to be recharged frequently (so big "recharging farms" have to be planned if a whole class is to be equipped regularly), and it is not possible to use them in two consecutive lectures. However, we expect the battery lifetime of PocketPCs (and that of notebook PCs as well) to become longer soon, following a trend similar to that of mobile phones. For example, modern notebooks or TabletPCs with Centrino mobile technology can be expected to run up to 4 hours, thus more than twice as long as a typical notebook with similar load runs today.

The fact that fewer standard applications are available for the PDA than for the notebook played no role in any of the scenarios, neither in those in the literature nor in our own. The reason is that most scenarios use either a web browser (available on

all devices) or specifically developed software for quizzes, online feedback, etc. We strongly favour portable software (such as our WIL/MA software written in Java) what will run on all modern mobile devices.

The really relevant differences between notebook, TabletPC and PocketPC are screen size, weight, and input devices: Whereas multiple choice questions, other types of simple quizzes or the feedback tool work very well on the small screen of the PDA, interaction metaphors demanding high-resolution graphics or long texts require the larger screen of a notebook or tablet PC. Annotations on transparencies always require much screen space; permanent scrolling on the small PDA screen proves to be cumbersome. When creating new applications for educational scenarios user interfaces for devices with limited resolutions are usually much harder to design than those for bigger screen sizes.

As far as the size and weight are concerned, the results of our four extensive studies show that individual students have strong preferences; but we observed no clear trend. The reasons to prefer the PDA were usually:

- it is smaller and does not occupy as much table space as a notebook (e.g., so the student can still use a printed script),
- it is lighter than a notebook, so not much extra weight has to be carried around,
- it is not as distracting as a notebook because only limited software is available,
- you don't have to look over the screen to see the lecturer,

whereas the reasons to prefer the notebook were:

- you can use the notebook to annotate electronic scripts,
- the screen space is much bigger, so the software is easier to handle (and to see, especially for students with impaired vision),
- you can use the notebook for other things beside the lectures (receive and send emails via wireless LAN, etc.).

Another major difference is the input metaphor. Notebooks use a mouse - or a mouse replacement, e.g., touchpad - to control the graphical user interface and a keyboard to enter text. Most PDAs neither have mouse nor keyboard, the user can

draw directly on the screen with a stylus instead. While this input type is much more intuitive than the mouse for pointing and clicking, it is almost impossible to enter larger portions of text with it. The user can choose to use either a virtual keyboard, graffiti or hand writing recognition. The virtual keyboard is displayed on (and consumes most of) the screen and is operated by pushing the virtual keys with the tip of the stylus; this is in our opinion the easiest mode for inexperienced users. More practice is needed to use graffiti: the user draws each individual character in a certain way as a single stroke in a specific area of the screen. Although stated differently in advertisements, the third alternative (trying to let the device recognise handwritten words or text pieces) doesn't work at all for most users.

The consequence is that if a lecture scenario depends on the students' ability to enter more than a few words, PDAs should not be considered. On the other hand, PDAs are a better choice if the students are supposed to draw something (a graph or mathematical formulas) on their screen. Tablet PCs are an option in between, offering stylus-based input on the screen, mouse-like input and a keyboard. In our experiments, PDAs and their stylus-based user interface were perfectly suited for the interactive lecture.

Last but not least, there is a huge difference in costs for these devices. Notebook prices are ranged between 600 € and 2500 €, TabletPCs are no less than 1000 € at the moment. PDAs on the other hand are sold for less than 400 € including WaveLAN support[2].

6.2.3 Installation and Configuration

The small, lightweight computers combined with wireless connectivity allow a very easy setup for the WIL/MA tools. For the WIL/MA infrastructure, we have repeatedly used the following configuration:

- A mobile access point (AP), compatible to IEEE 802.11b or higher, is used to establish the local area network the clients can connect to. In our experience, cheap consumer devices are sufficient. If the lecture hall has wireless LAN

[2]the prices were taken from the web-sites of major European hardware vendors in January 2005

installed, this may be used instead. Usually, the WIL/MA LAN is not connected to the Internet to avoid distraction.

- A server notebook that is powerful enough to run both the server and the administrator client. For a normal sized class (up to 100 students), a standard notebook is adequate as long as a network-capable operating system (Linux, Windows XP Professional, etc.) is installed. This notebook is best connected to the AP directly with a cable (Ethernet) to save precious wireless LAN bandwidth.

- Optionally, a teacher notebook may be required. If the teacher needs assistance for the interactive lecture, the assistant will use the administrator notebook to control the services. The teacher then needs a console to get access to the relevant data. A TabletPC is the perfect device for this task for its easy handling, but a lightweight notebook will also be sufficient. This computer does not have to be connected to the AP by means of Ethernet; wireless LAN can be used just as for the student clients.

All required hardware and an additional multi-outlet power strip fit easily into a notebook travel bag. With proper preparation and a little bit of exercise, it takes only about 3 minutes to install the complete setup.

Much more time is needed to hand out the mobile devices. For our first experiments with notebooks, we had to allocate the lecture hall for an additional hour before the lecture started to install all the necessary power strips and computers and to prepare them for the students. Fortunately, it is much easier with PocketPCs.

Before the course starts, the students have to prove their identity and register with their full name, matriculation number and email address in order to get a "WIL/MA Card" with a unique serial ID. During the break before the lecture, the students can exchange their card with a PocketPC; usually two or three teaching assistants are responsible for a smooth progress. After the lecture, they return the PocketPC and get their card back. With our pool of 70 PocketPCs, both activities take up less than ten minutes and are thus well within the limits of the usual break between two lectures.

In the fifth experiment, we tried to lend the PocketPCs to the students for the complete semester, but this did not work out very well. First of all, many students did not use their device regularly: 50 devices were handed out but only about 30 were used in each lecture. Most PocketPCs hadn't been charged properly before the lecture, so the battery did not last until the end. The biggest problem, however, was to retrieve all devices. While most students returned their PocketPC reliably, we had serious problems recovering the remaining few in due time before the next experiment started.

6.3 Evaluation Results

In this section, we will present a selection of important results of all our experiments in Mannheim. Note that for better comparability the values of the surveys have been recalculated to a percent scale with 0% meaning "I do not agree to the item at all" and 100% meaning "I do fully agree to the item". The actual values were measured with different scales. Also, the questionnaires and the remarks from the students were usually in German and were translated to English for the purpose of this dissertation.

6.3.1 Acceptance of the Interactive Lecture

The acceptance of the interactive lecture was generally very high, regardless if it was tested with students in a technical or a liberal arts course. In the first experiment, the students were downright enthusiastic about the new possibilities, so we had to impute at least some of the praise to the novelty of the scenario. In the next experiment, the differences between the acceptance of the traditional lecture and the interactive lecture were more realistic: Although the traditional part itself was rated very high, the interactive phase of the lecture achieved an even better score. Given the fact that both phases contained eight full lectures and that the novelty effect can thus be considered to be of no relevance, we can claim with high significance ($p < 0.001$[3]) that

[3] $p<0.001$ means that with a chance of only 0.1% this hypothesis is actually wrong

Item	**traditional**[1)2)]	**interactive**[1)2)]	**significance IL>TL**[3)]
I am content with my learning success today	**65.3%** *(23.3)* **53.0%** *(23.7)*	**93.3%** *(13.7)* **70.0%** *(23.3)*	$p < 0.001$ $p = 0.034$
The lecture today was diversified	**43.3%** *(29.0)* **45.0%** *(23.3)*	**86.7%** *(17.0)* **84.0%** *(22.7)*	$p < 0.001$ $p < 0.001$
I had the opinion that I could participate actively	**29.3%** *(25.3)* **23.7%** *(25.7)*	**75.7%** *(34.3)* **71.3%** *(32.0)*	$p < 0.001$ $p < 0.001$
I attended the lecture particularly intently today	**56.7%** *(25.7)* **49.0%** *(29.0)*	**86.7%** *(21.0)* **74.7%** *(18.0)*	$p < 0.001$ $p = 0.002$
I have learned more today than in other lectures	**40.7%** *(23.3)* **33.3%** *(31.3)*	**77.7%** *(30.0)* **65.0%** *(29.7)*	$p < 0.001$ $p = 0.003$
The lecture was generally very good	**69.0%** *(15.7)* **62.7%** *(20.0)*	**91.0%** *(15.3)* **79.3%** *(26.7)*	$p < 0.001$ $p = 0.041$
sum of all 14 items	**58.5%** *(10.4)* **52.8%** *(14.6)*	**79.2%** *(13.6)* **72.0%** *(13.8)*	$p < 0.001$ $p < 0.001$

Notes:

[1)] For statistical correctness the two groups are accounted for separately: the top value in each cell belongs to group 1 ("operating systems"), the bottom value to group 2 ("image analysis").

[2)] The numbers in the cells are the **mean value** *(standard deviation)* of the sample

[3)] Hypothesis that the interactive lecture (IL) is rated better than the traditional lecture (TL); p is the chance, that the assumption is not correct.

Table 6.2: Acceptance ratings for the interactive lecture in the try-out 2001/02, selected items

the students indeed like the idea of an interactive lecture. The original values of the two early experiments are listed in Tables 6.2 and 6.3 with a graphical visualisation in Figure 6.7.

High acceptance ratings were also obtained throughout the following experiments. Furthermore, we received very encouraging comments in surveys or group discussions.

Item	**traditional**[1)2)]	**interactive**[1)]	**significance IL>TL**[3)]
I am content with my learning success	**77.7%** *(19.3)*	**87.3%** *(13.7)*	p = 0.030
The lecture was diversified	**65.0%** *(24.7)*	**77.7%** *(22.0)*	p = 0.072
I had the opinion that I could participate actively	**38.0%** *(19.0)*	**66.7%** *(25.8)*	p < 0.001
I attended the lecture particularly intently	**70.0%** *(20.7)*	**76.0%** *(18.7)*	p = 0.162
I have learned more than in other lectures	**62.0%** *(26.3)*	**71.3%** *(19.0)*	p = 0.162
The lecture was generally very good	**76.3%** *(18.7)*	**85.7%** *(16.7)*	p = 0.083
sum of all 16 items	**72.0%** *(11.3)*	**80.0%** *(08.7)*	p = 0.005

Notes:

[1)] The numbers in the cells are the **mean value** *(standard deviation)* of the sample

[2)] The survey was carried out after the first traditional and before the interactive phase.

[3)] Hypothesis that the interactive lecture (IL) is rated better than the traditional lecture (TL); p is the chance, that the assumption is not correct.

Table 6.3: Acceptance ratings for the interactive lecture in the second study 2002, selected items

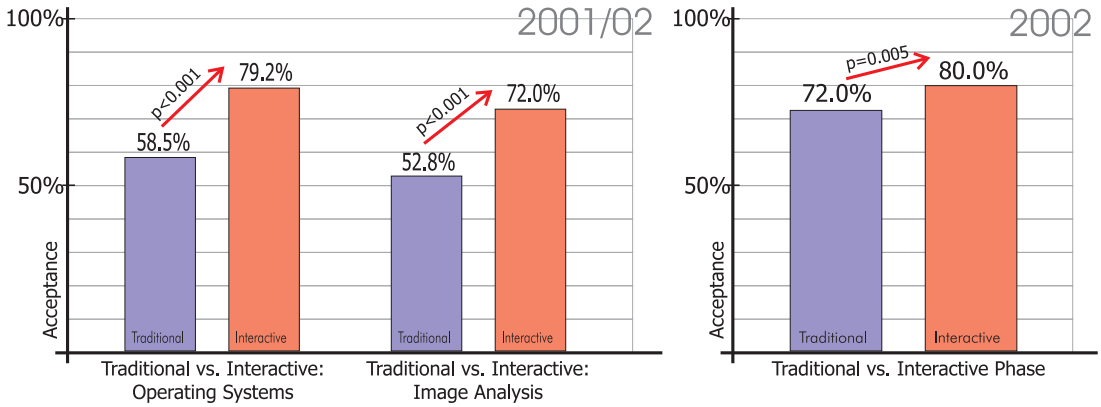

Figure 6.7: Acceptance ratings in the first two studies

A selection of positive, but also some critical remarks are listed below:

- "Should be used in more courses" *(Jan. 2002)*
- "The laptops should be used for more activities, i.e., feedback or to ask questions" *(note: this remark was received after the try-out in January 2002 where only the quiz service was used)*
- "Best lecture of the whole semester, should be repeated and continued, very good!" *(Jun. 2002)*
- "Using the devices is a good idea but the lectures should not be overrun because of that" *(Jan. 2002)*
- "The call-in service is good but definitely a distraction because the input of the question takes too much time, but I don't know a feasible alternative..." *(Jun. 2002)*
- "The interactive quizzes are very reasonable and kept me awake even in the evening." *(note: the lecture in June 2002 was between 3:30 and 5:00 PM).*
- "The computers are too distracting to be useful. Even if you don't use a computer yourself, you feel distracted by other students playing around..." *(Jun. 2003)*
- "I think the concept of interactivity is a very good idea" *(Jan. 2004)*
- "Computer networks was definitely a lecture of the future –>more of that!!!" *(Jun. 2003)*

In our fourth study, being the first study with students from a non-technical department, we asked the students about their preferences regarding the inclusion of computers in a lecture at the end of the course. When asked which form of participation they would prefer, 65.3% stated that they would like to use an electronic device[4] and only 18.9% would rather use paper and pencil instead. We also wanted to know if the students would like to participate with mobile devices in future courses: 69.5% would repeat this experience and only 9.5% prefer a traditional lecture. If they had to choose between the same course offered both interactively and traditionally, 82.1% of all asked students would participate in the interactive course. Last but not least,

[4] 48.4% were content with a PocketPC, 16.8% would like to have a more powerful device, such as a notebook

the very provocative question if the students would appreciate all future courses to be interactive was approved by 31.6% and rejected by only 22.3%.

Actually, there was very little difference at all between the students from different departments regarding acceptance. As an example, the ratings of the quiz service as one of the most thoroughly investigated services from three different experiments are summarised in Table 6.4 on the facing page. The psychology students were only slightly less positive about the quiz service than students in the computer science course. The rather high distraction felt by the students was partly due to the concurrent evaluation, as we discovered in several remarks from the students.

6.3.2 Acceptance of the Tools

Of course, we also wanted to know how the students rate the tools in order to be able to improve them in the next release. Therefore, we added a hardware evaluation section to the questionnaires at the end of the semester in all but the first experiment; in Table 6.5 on page 116 the results of the ratings are shown.

As you can see, the grades given by the students for several attributes of the tools are generally very good. There was a slight deterioration when we started to use the new, revised user interface on PDAs for the first time in Summer 2003. This was due to some problems with the first versions and - primarily - the deficits of the PDA compared to a notebook in terms of display size and speed. Over time, we managed to significantly improve the technical reliability as well as the user interface and have now almost achieved the rating the tools had when they had been adapted to the notebook's screen size and input paradigms.

6.3.3 Effects of the Interactive Lecture

Some of the experiments were also designed to measure various effects of the interactive lecture to see if we had actually managed to improve the lecture as an educational setting, as discussed in chapter 2.4 on page 20.

Item	**Summer 2002**[1)2)] comp.science UCE-Tools	**Summer 2003**[1)2)] comp.science WIL/MA	**Winter 2003**[1)2)] psychology WIL/MA
The textual design of the quiz service is intelligible	**51.6%** *(11.2)*	**84.6%** *(19.0)*	**76.0%** *(23.8)*
The handling of the quiz service is comprehensible	**53.0%** *(09.8)*	**89.2%** *(18.0)*	**86.2%** *(19.6)*
Using the quiz service has distracted me from the lecture	**40.0%** *(20.6)*	**50.0%** *(31.6)*	**45.8%** *(28.4)*
By using the quiz service, I could concentrate less on the topics of the lecture	**46.4%** *(16.6)*	**54.6%** *(32.8)*	**47.8%** *(29.0)*
I would recommend the quiz service to other students	**49.2%** *(14.6)*	**68.4%** *(33.0)*	**60.0%** *(23.6)*
I think the quiz service is well done	**48.2%** *(12.4)*	**72.2%** *(27.2)*	**64.4%** *(22.0)*
I rate the quiz service with the school grade...[3)]	**2.10** *(0.98)*	**2.26** *(1.15)*	**2.41** *(0.85)*

Notes:

[1)] all values were taken from the last survey of each semester

[2)] The numbers in the cells are the **mean value** *(standard deviation)* of the sample

[3)] Original values using the German school grade system: "1" being "excellent" and "6" being "insufficient"

Table 6.4: Acceptance ratings for the quiz service from three experiments.

Item	**Summer 2002**[1)2)] comp.sc. UCE Notebook	**Summer 2003**[1)2)] comp.sc. WIL/MA PDA	**Winter 2003**[1)2)] p.psych. WIL/MA PDA	**Summer 2004**[1)2)] comp.sc. WIL/MA PDA
General handling	**2.07**	**2.20**	**2.19**	**2.03**
User interface design	–[3)]	**2.44**	**1.96**	**2.16**
Menu navigation	**1.90**	**2.28**	**2.07**	**2.06**
Clarity of the display	**1.55**	**2.24**	**2.02**	**2.25**
Quiz service: design	**2.00**	**2.24**	**2.07**	**1.73**
Quiz service: handling	**1.93**	**2.04**	**2.16**	**1.77**
Call-In: design	**1.86**	–[4)]	–[4)]	**2.40**
Call-In: handling	–[3)]	–[4)]	–[4)]	**2.71**
Technical reliability	**1.90**	**3.28**	**3.22**	**2.51**
Speed	–[3)]	**2.38**	**2.21**	**2.38**
Overall score	**1.89**	**2.38**	**2.27**	**2.16**

Notes:

[1)] The numbers in the cells are the **mean value** *(standard deviation)* of the sample

[2)] Original values using the German school grade system: "1" being "excellent" and "6" being "insufficient"

[3)] This item was not inquired at this time

[4)] The Call-In service was not used in this experiment

Table 6.5: Acceptance ratings for the WIL/MA tools in several experiments.

Improved Motivation

To measure the motivation and attention, we used online surveys during the lectures, assignments after the lectures and group discussions. Although we did not succeed to prove a solid motivation gain, there are several indications that support this hypothesis. The online surveys, though not yet fully analysed, had five measuring times each throughout the lecture. When plotting the average attention levels over a traditional lecture (Summer semester 2002), we get a curve bearing a strong resemblance to the attention trend measured by McLeish (see chapter 2.2.3 on page 14). In an interactive lecture, however, the trend does not degrade that much and stays generally on a higher level.

Also, when asked directly, the students stated that they were more attentive during an interactive lecture. They claimed that the regular quizzes effectuated a change of activities which most students felt very positive about. More than that, many students tried to do well in the quizzes, so they focused more on the lecture and tried to figure out which questions may be included in the next quiz. Particularly advanced question types like the "clickable image" used sporadically in the last two experiments were considered exceptionally valuable. Unlike multiple-choice questions they do not offer ready-made answers for the students to choose from, but instead force them to deduce the correct answer on their own and thus to deal with the question more intensively.

Both other services do not seem to have an effect on the motivation. The call-In service, however, is successful nonetheless: about a quarter of all students equipped with a mobile device used the service regularly (about once a lecture). This is a multiple of the usual participation with traditional means.

Furthermore, we could not observe an effect on self-efficacy[5]. Although tested twice with a number of dedicated tests, we were not able to get significant results, so neither a positive nor a negative effect could be deduced.

[5]Self-efficacy is the general belief of a person to see the source of events in him- or herself (as opposed to consider external circumstances or other people to be responsible). In learning environments, self-efficacy means that a student believes to be able to learn and understand a certain topic autonomously.

Learning Success

Along with a higher motivation and attention we anticipated a higher learning success. Unfortunately, this could not be proven with the first experiment. With high significance ($p<0.001$), we could show that the students did indeed learn something in any lecture, but the difference between the learning success in an interactive or traditional setting was only marginally higher (in favour of the interactive lectures), and insignificant. This, however, could at least be used as proof that the "technical knickknack" (as it was called by a less enthusiastic student) did not distract the students excessively.

The second study had a much more elaborate pre and post knowledge test and compared the two settings over a total of 16 lectures. Again, a learning success could be verified with high significance, but this time, a positive effect of the interactive lecture could be demonstrated, too. With a convincing significance ($p<0.001$), the students learned noticeably more in the interactive phase than in the traditional phase before. The results of both learning success measurements are visualised in Figure 6.8.

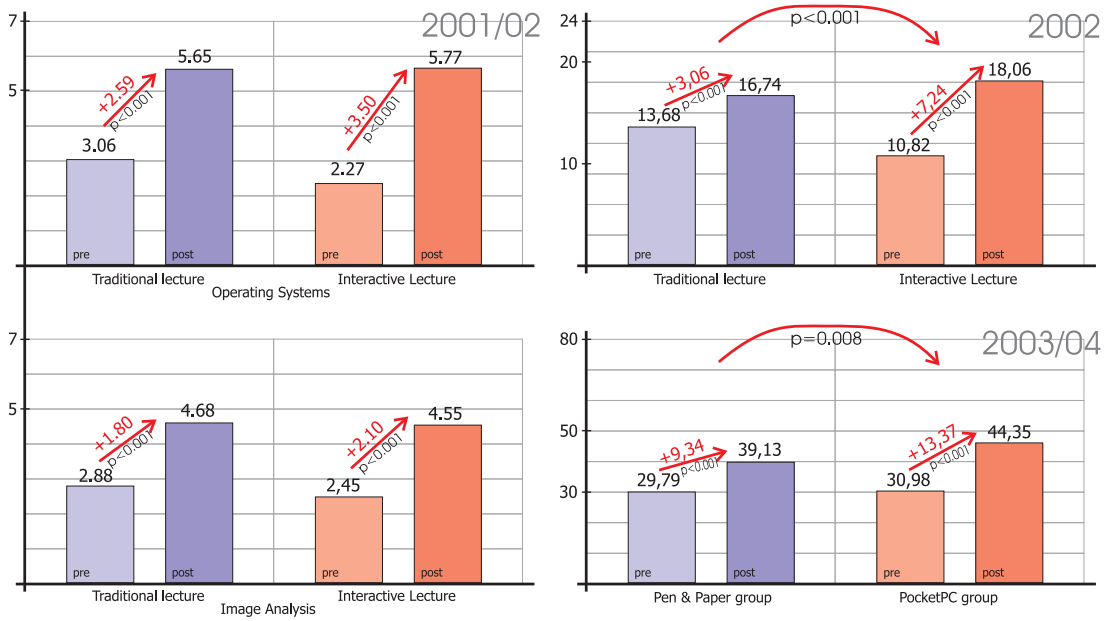

Figure 6.8: Learning success measurements in three different studies

Computer Familiarity

An interesting side effect of the integrated utilisation of PocketPCs, the need to register by means of email and several web-based questionnaires could be observed in the 4^{th} experiment. Most of the students in the PDA group who usually had little experience with computers before, claimed that they felt more confident in the use of computers than they did before the interactive lecture. This effect, of course, could not be repeated with students of the technical department.

Still, we can conclude that it is not only perfectly possible to apply interactive lectures in non-technical departments, but it may also have a positive impact on the training of soft skills.

6.3.4 Some Comments on the Quiz Service

One of the problems we had been confronted with was the question how many quizzes should be inserted into a lecture and how many questions should be asked in a quiz. Depending on the difficulty of a question, the students should have about one to two minutes for the answer. Another minute should be added per quiz to give the students some time to get an overview. The proper discussion of the quiz results takes another one to two minutes per question.

Therefore, the typical quiz with three questions uses up about 10 minutes of the lecture time. In our experience, three quizzes of this size are too much for a 90 minute lecture, but two quizzes seem to be a very reasonable choice. Alternatively, we had tried three quizzes with two questions each. While this shortens the time between the discussion of the topic and the exercise and allows the teacher to organise the interactive lecture more dynamically, more time is consumed in this way.

In the 4^{th} experiment, we also asked the students for their opinion regarding the number of quizzes per lecture. Based on a three questions per quiz setting, 63.7% of the students were content with two quizzes, 29.4% would like to have more quizzes and only 6.9% think that a single quiz per lecture would be sufficient.

In this context, we also investigated the possibility to save some time on the discussion of the results. Therefore, we tried three different kinds of feedback in the third

experiment: first we started with no discussion (only the result graph was displayed for a few minutes), then the lecturer discussed the correct answers only but did not explain why the other answers are wrong. Last but not least, all answers were discussed in detail. The results can be seen in Figure 6.9: on the left side, the acceptance ratings for the three feedback variations are sketched and on the right side you can see the learning success in the three phases.

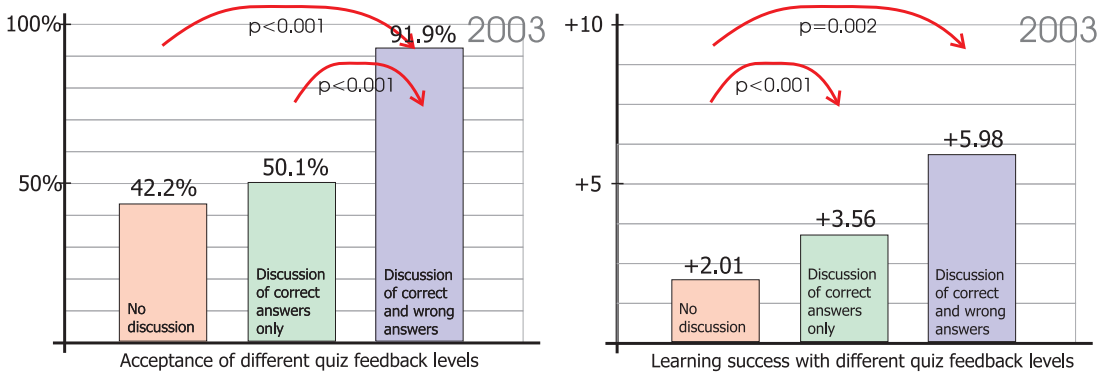

Figure 6.9: Acceptance and learning success in correlation with the detail level of the discussion of quiz results

With high significance ($p<0.001$), the students rated a thorough discussion higher than the shorter versions; between these, however, there is no discernible difference. This also had an effect on the learning success. In the variations with at least some discussion, the learning success was significantly higher ($p<0.001$ for variation 2 vs. 1, $p=0.002$ for variation 3 vs. 1) than without feedback. As a result we can derive that the quiz is most useful only if the lecturer also takes some time to explain the answers properly.

Although investigated thoroughly in the 4^{th} experiment, no statement about the quality of the different types of personal feedback can be deducted: Neither the individual benchmark, the social comparison nor even omitting all feedback had a clear advantage over the other alternatives.

6.4 Experiments Outside of Mannheim

The WIL/MA software has not only been used in Mannheim. Other German and also international universities downloaded the freely available software to implement it in their own courses, amongst these reportedly the University of Cottbus, the Technical University of Berlin and the Washington State University. Regular email requests and the statistics of our project web site propose even more users: An average of about 100 page hits per day and approximately 10 software downloads per month testify an active interest in our interactive learning scenario.

Beyond that, our intentions to collaborate with other universities resulted in two joint projects with institutes at Stanford University. While our experiments in Mannheim concentrated on the key features and characteristics of the interactive lecture, these two additional experiments had other objectives. In both studies, our software was altered and used intensively.

6.4.1 WIL/MA in a global A/E/C course

The PBL Lab (Problem-Based Learning) is the home of an integrated research and curriculum development effort launched in 1993 in the Department of Civil and Environmental Engineering at Stanford University and directed by Dr. Renate Fruchter. The goal of this lab is to enhance global teamwork with several partners world-wide. One of their key activities is the yearly A/E/C course. In this course, three to five teams made up of architecture, engineering and construction management students are given realistic projects from partners in the industry. The students in a team are distributed world-wide; the architect may be a student in Germany while the engineer is Japanese. Nonetheless, they have to design a building's interior and exterior, choose appropriate materials, plan the construction time table and stick to the budget.

At the beginning and the end of the 6 months course, they meet in person in Stanford. At all other times, they have to use video conferencing and other collaboration tools to work on the project. During the course, all teams have to present their

preliminary results in two large conference sessions and evaluate and discuss the presentations of their fellow teams. The evaluation process during these sessions was to be improved with WIL/MA.

The students from three teams volunteered to participate in this experiment. They installed the WIL/MA student client on their notebooks (which they needed for their presentation anyway) and used it at certain times during the conference to give their feedback, while the other teams were asked to evaluate the previous presentation traditionally. Although originally intended for continuous use, the feedback service seemed most appropriate for this purpose. Three categories were introduced: "How valuable was it for you to observe the project crit of the [name of one of the other teams] team?", "How relevant was the industry mentor input for your project?" and "How relevant were the team solutions to your project?" which could be rated from 1 ("not at all") to 5 ("exceedingly"). Since the participants were not physically present, the connection between the clients and the server was established using the Internet instead of a local wireless LAN.

Although the sample was very small, it could be observed that the student's feedback with WIL/MA was much more discerning than without the tool: The students rated the efforts of the other teams lower in the pseudo anonymous environment created with the WIL/MA tools than when asked to give their feedback in person. Furthermore, the feedback was instant, while without WIL/MA each team has to be called one after another, which takes a lot of the precious time. A screenshot of the responses of the German team (Bauhaus university) is shown in Figure 6.10 on the facing page.

6.4.2 The CodeBreaker Curriculum

While in the PBL experiment a standard version of WIL/MA was used, the second experiment, designed by Professor Roy Pea and a number of graduate students at the Stanford Center for Innovations in Learning (SCIL), required much adaptation. The study took place at a high school in Santa Clara, about 30 miles south of Stanford. Four summer school math classes with around 25 thirteen-year old students each were equipped with PocketPCs (first and second class) and TabletPCs (third and fourth

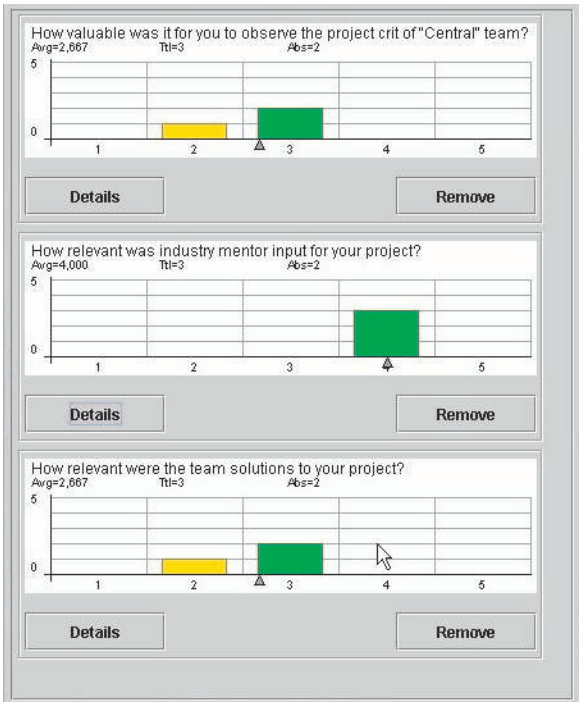

Figure 6.10: The results of an A/E/C evaluation

class). For four weeks, the students worked in small groups together on a special math curriculum using their mobile computers intensely.

The curriculum ("CodeBreaker") had been created by professor Shelley Goldman (Department of Education at Stanford University) some years ago [Gol01]. It is supposed to give a playful introduction into mathematical functions: the kids are given a text, and by assigning a number to each character ("a"=1, "b"=2, ...) and using a simple mathematic formula of the type $f(x) = ax^y + b$, they "encode" this message. The encoded message (a row of numbers) is given to another group that tries to find out the "key" (i.e., the three parameters a, y and b used in the "encryption" formula) to retrieve the original message. A schematic diagram of the students' tasks is sketched in Figure 6.11 on the next page.

Originally, the curriculum was designed as a pen-and-paper course with very detailed guidelines for teacher and students. Because it is very cumbersome to recalculate the character-number translation tables each time a parameter of the function is changed, a supporting computer program for Apple Macintosh computers had been written. This, however, had the problem that only one computer per team proved

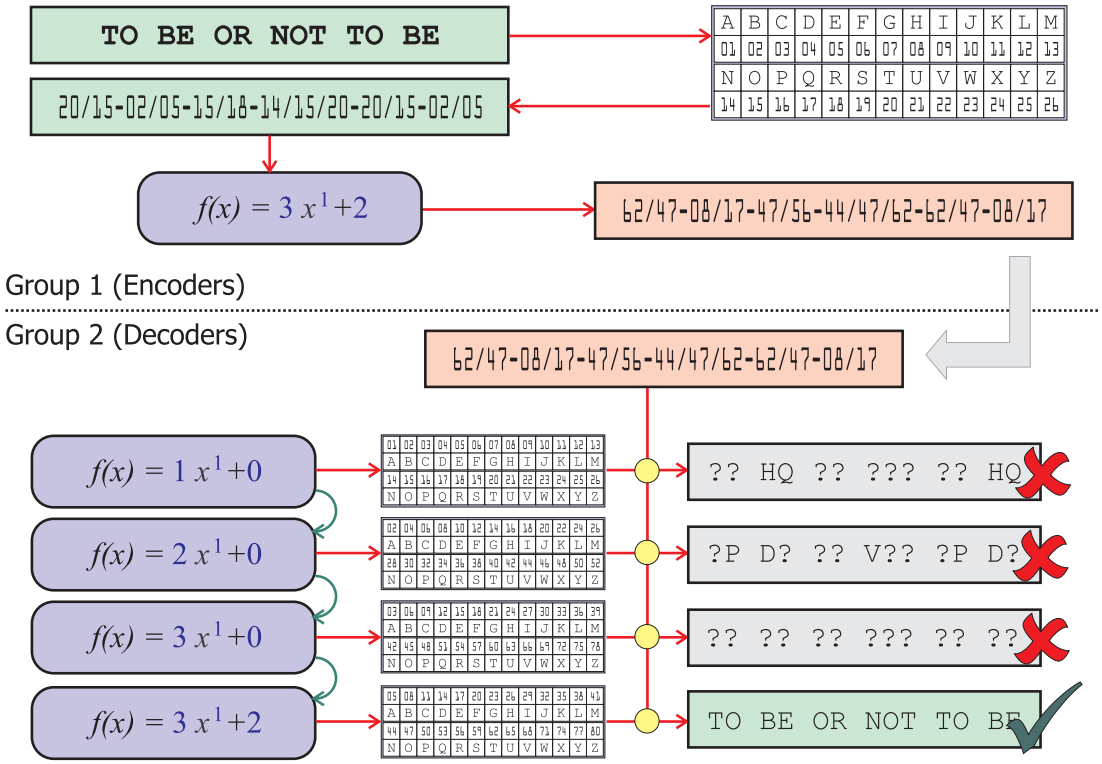

Figure 6.11: The scheme of the CodeBreaker! curriculum: Coding and decoding text messages

to be a bottle-neck: active kids battled for the place in front of the computer while passive kids were pushed into the role of an observer.

With WIL/MA, a third variation of the curriculum was tested. Each student was equipped with his or her own computer, but still the teams were supposed to work on common tasks. To do this, the WIL/MA toolkit had to be extended with a CodeBreaker module as well as an early group support mechanism with which a teacher was able to assign the students to different teams. Between team members, additional update packages were exchanged to synchronise the computers in a group: most actions performed by a student had an impact on the common state and were thus visible to the team members as well.

In Figure 6.12 on the facing page, this behaviour is demonstrated with a number of screenshots. Two students, Peter and Kevin, just started the software. While Kevin is already assigned to the Group “Group2” by the teacher, Peter is unable to do something because he has logged in for the first time. As soon as the teacher has

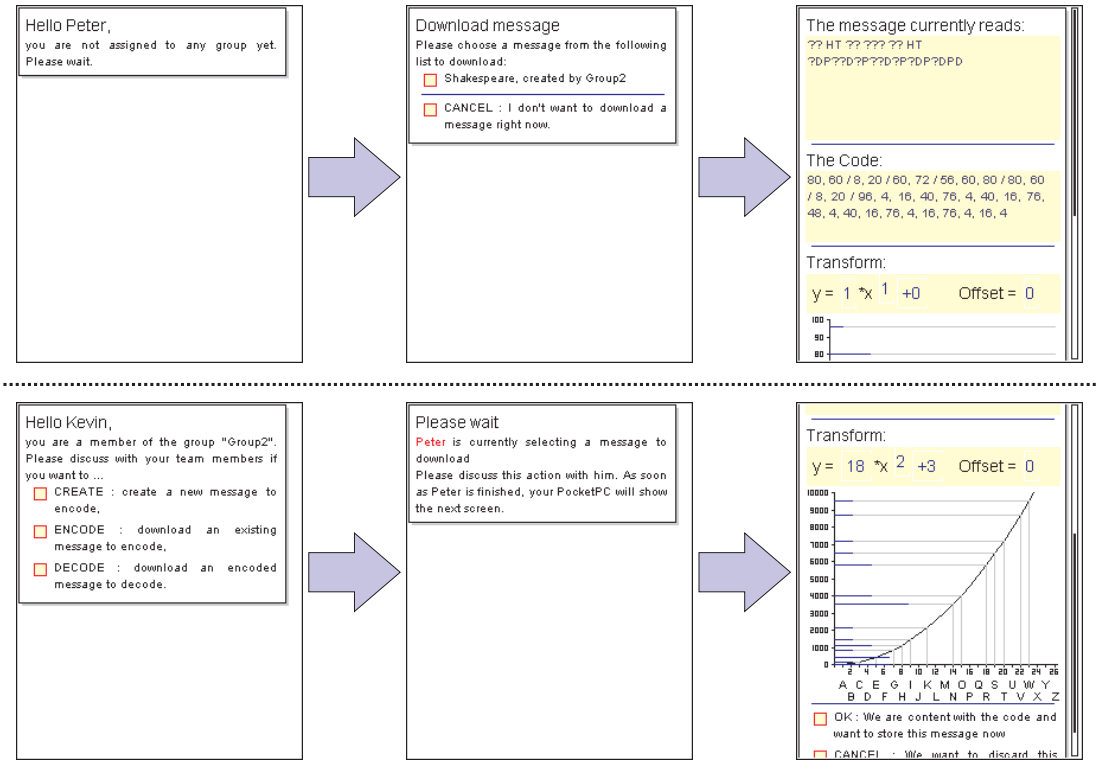

Figure 6.12: Screenshots of the CodeBreaker! student client showing different states of two team members

put Peter into the same group, he will also see the start menu. Peter then decides to download a message from the server to start with decoding. While he is doing so, Kevin's client is in a wait state; only one student at the same time can select the next cipher text for the group. When the message is loaded, both Peter and Kevin can act again. Their task is to find the correct parameters for the method to decipher the message. Both kids can change the parameters, but each change is also propagated to the team member, so they have to agree on their actions verbally. Their advantage, though, is that they can both use different views on the task; in this example, Peter looks at the partially deciphered message while Kevin observes the function graph. To maximise the experience, there were a number of other visual aids, too, like a character table and a frequency breakdown.

The teacher's client was implemented as a graphical user interface started directly by the CodeBreaker server module because it was too complicated to be incorporated into the administrator client framework that was still at a very early stage at this time. With the CodeBreaker control application, the teachers can control the curriculum

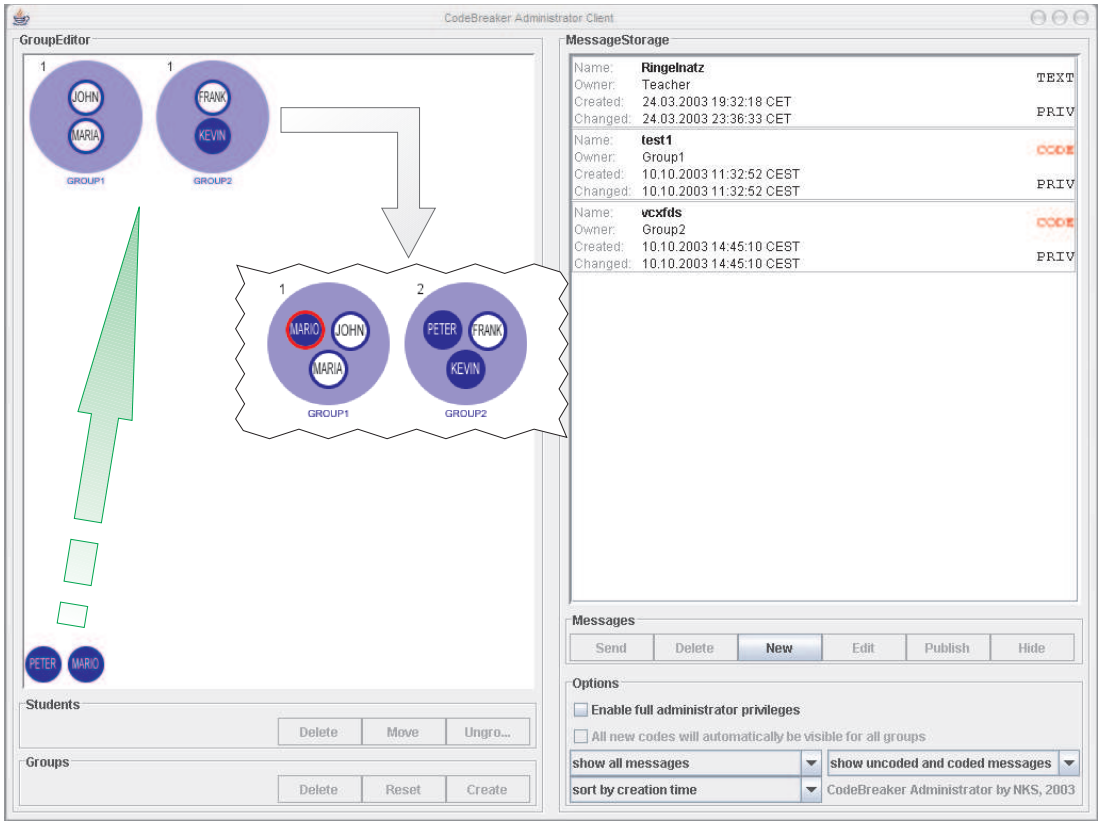

Figure 6.13: A screenshot of the CodeBreaker! teacher client

in detail, see Figure 6.13 for a screenshot:

- Groups can be created (displayed as light blue discs), named and deleted.

- Students can be assigned to groups, removed or reassigned at any time. They appear as dark blue circles with the name of the student. Icons of not yet assigned students are put at the bottom of the group builder. With drag-and-drop, they can be moved to a group icon; the icon then is displayed inside the group sphere.

- The group builder also gives a permanently updated overview of the students and the groups: students with inactive clients (lost connection) are displayed as white discs with blue border; the current group state (choosing a message, encoding, deciphering,...) is displayed as a number in the top left corner of the group icon.

- Messages can be created, viewed, prepared and centrally assigned to the students.

- "Spies" can be launched; these are little windows showing the contents of the student clients of a particular group. For the team members of that group, however, the spy remains invisible. This feature can be used by teachers to observe the actions of the students in detail.

- The computers of one or more groups can be remotely controlled to a certain extent.

The implementation of these extensions took place at the SCIL in order to be able to discuss the realisation of the project on site; our stay was kindly supported by Stanford University and financed by the Learning Lab Lower Saxony (L3S) as a partner of the Wallenberg Global Learning Network (WGLN). While our part was to provide WIL/MA as software basis and to develop the administration and communication processes, the logic of the curriculum and the user interface metaphors were mostly implemented by John Murray, software developer at SCIL.

During the experimental phase, almost every bit of data was recorded: Three video cameras per class filmed the students and the teacher, microphones recorded the discussions of three teams per class. The journals of the pupils were copied and all actions with the mobile devices were logged. The first results of this experiment were published in [GPM04a] and [GPM$^+$04b] but a vast amount of data is still under evaluation.

The most important conclusions so far can be categorised in the following way:

- The learning success of the students is very promising, considered that the experiment took place in a Summer school course for students who had severe problems with learning and understanding math.

- The free role model of the software did not work out very well, strict roles had to be developed and assigned to single group members in the course of the experiment to prevent "cursor wars".

- The teachers approved the technology, though it was very important to train them to use the software appropriately. After that, they developed their own ways to use the technology, adapting it to their individual teaching style.
- Technology shortcomings were handled very well both by the teacher and the students.

In summary, the experiment was very successful and since most of the students had good results in the post-tests, it can be assumed that our kind of learning scenario can be used reasonably in K-12 as well. For future experiments, the WILD@Stanford-team plans to expand the software with flexible grouping mechanisms and stricter role models; also the user interface, that was considered less than ideal, will be redesigned in the near future.

7 Support for Collaborative Learning

In this chapter, we discuss support for collaborative learning. First, a short introduction into the psychological background of team work in education is given, then a possible realisation for team support in an interactive lecture is presented, considering a modified quiz service as example. Last, but not least, participatory simulations as a promising new form of teaching scenario are discussed.

7.1 Pedagogical Introduction

WIL/MA was designed as a 1-to-n-to-1 communication between individual students and the teacher. Support for an uncontrolled communication amongst students (e.g., chats or SMS) was deliberately omitted in the software releases to keep the distraction level as low as possible for the students. This, however, means that WIL/MA indeed enhances interaction and for this reason motivation of the single students (as can be seen in chapter 6.3 on page 110), but it does not change the fact that the students, even in an interactive lecture, only learn for and by themselves, and not in groups.

Team work, however, is known to have strong motivational effects on learners and to consolidate the new-won knowledge. Many models of human learning propose a phase, where the learners have to discuss the current topics (the knowledge gained in prior phases) and thus reflect on them [Wei98][MCTR94]. This not only helps to memorise the subjects more profoundly but also to improve the ability to apply this (otherwise theoretical) knowledge. Furthermore, group work is highly accepted by students [KPCB02][BBC00].

To be most effective, though, team exercises have to meet certain demands [MMP73]. A comprehensive list of these demands was assembled by Slavin [Sla80]:

- A *common target* has to be specified for the team members to identify with. This motivates the students to be a part of their group and to work jointly on the solution of a given problem.
- *Individual liability* is important to eliminate the problems of "lurkers", i.e., team members with no or only very little contribution to the common target. The individual performance of each team member has to be noticeable.
- In this context, *skill interdependencies* inside a group are also desirable. This means that the team members are reliant on the skills and proficiencies of other team members and that each team member is able to deliver valuable input.
- Only *real teamwork exercises* should be used, i.e., problems that can not be solved properly by an individual alone.
- The exercises should have *no preexisting solution* that is obvious to the team. Furthermore, they should be interesting and exciting to intrinsically enhance the motivation of the students.

According to these recommendations, we investigated the possibilities of team work in interactive lectures and other similar settings.

7.2 Group Support in WIL/MA

Based on our experiences with the CodeBreaker experiment in Santa Clara (see chapter 6.4.2 on page 122), our first step was to include generic group support into the WIL/MA server core. Therefore, we identified the different levels of group support that are possible in this scenario.

A very common support for small groups in several related projects (e.g., Concert-Studeo) is to let students share a single mobile device. While this certainly helps to equip large classes with computer access for an interactive lecture, it does not improve team work. Furthermore, a number of problems is related to that feature; most of them had not yet been solved satisfactorily (e.g., unintentional glance at the quiz answers of fellow students, biased feedback, switching between the users being awkward or unsafe). Clinging to our belief that in the near future most students will

own a WIL/MA capable device anyway we intentionally abstained from including device-sharing support.

A second approach that also has been used in prior projects (ClassTalk, Discourse) is based on a similar idea: again, a group of students is equipped with a single computer, but in this scenario the members of the group are supposed to work as a team. This, however, has some serious disadvantages: first of all, most interactive services (i.e., all services that can be used continuously) are unfeasible in this scenario, leaving only a number of asynchronous services like the quiz. Then, the teacher has only very little control over the composition of the teams because usually a team will consist of students sitting together and knowing each other anyway. This may potentially lead to huge differences between "elite groups" and "loser groups" and encourage students to simply join the opinion of others instead of discussing the problem appropriately: neither individual liability nor the balancing of skill interdependencies is properly cared for. Also, it is not possible to mix teams and individuals in a class. Last but not least, students of a team have to be spatially close to each other to be able to discuss an exercise. This way, teams beyond the borders of the lecture hall (e.g., in tele-lectures) or randomised teams are not possible.

For these reasons, we have favoured the third approach: a support for teams where each team member has his or her own computer. Most of the problems just described are solved in this approach. The students can work in a team in certain activities (e.g., quizzes), but can act individually on all other services. Team members do not have to be placed next to each other; elaborate communication paradigms help scattered groups to work on their common target. This also means that the teacher has complete control over the groups; it may be even possible to keep the team members anonymous.

Of course, this approach creates new challenges. First of all, the messaging between the team members has to be carefully designed. Natural metaphors, like verbal conversations or facial expressions have to be mimicked, or, if this is not feasible, compensated for with visual aids only possible in computer supported settings. The goal is to design team work supporting services in such a way that scattered teams are not much handicapped compared to traditional teams.

Furthermore, various ways to form teams have to be evaluated to find out which one is most useful in a given scenario:

- Students can team up by themselves without being influenced by the teacher. To optimise their team, the students can look into tables with certain data, like previous quiz results, participation level or preferences of their fellow students. Teams are formed by inviting other students and by accepting an invitation.
- The teacher determines team leaders for the purpose of forming a team. Only the leaders have access to the information tables and can invite other students. The selection of team leaders may be random or deterministic (e.g., the best or weakest students). In this setting, the teacher has at least some influence on the formation of the teams.
- The teams can be determined solely by the teacher.
- The teams can be defined by the software automatically. This selection may be purely random (it may even be reasonable to randomise the groups anew for each exercise), but it is also possible to base the selection on various information that has to be gathered beforehand (e.g., knowledge tests or surveys). The latter approach was thoroughly discussed in [LE04].

7.2.1 Structural Expansions

Various enhancements to different parts of the WIL/MA software are necessary to realise a team support. The most important adaptation has to be done on the server core system, particularly on the user management and the messaging subsystem.

User Management

The user management has been upgraded with a dedicated group concept. A new server command allows invoking all methods needed for proper group control: groups can be created and deleted this way, and students can be assigned to a group, removed from a group or reassigned to another group at any time. Unlike other server commands, the access rights to the various subcommands are not fixed but can be changed by an administrator at runtime. This is useful, for example, if the students

are supposed to create groups and fill these groups only during a certain time span, but should not be able to change the group structure afterwards.

The group information is stored on the server's hard drive. Only the group affiliations of registered users are persistent, though. Anonymous or temporary users may be assigned to groups, but the assignments are cancelled when the user logs off or when the lecture has ended. On the other hand, it is allowed to modify the group structure at any time, even if one or more students are not logged in (or have never been).

Clients can access the group structure completely or partially with two additional subcommands. Relevant changes are propagated automatically.

Messaging

Team support requires many additional messages to synchronise or forward information between the team members. Therefore, the `MsgCommand` class was enhanced with three additional addressing formats. These formats can be used to send an update message to all users of a group, all but one user of a group and to all connected users but those of a particular group. These enhancements allow for a very detailed and efficient message distribution.

Additionally, methods were implemented to inform service modules about changes in the group structure each time a registered user joins or leaves a group. The modules are now also able to store personal information persistently, not only for individual students, but also for the teams. While personal user information is still only accessible through the corresponding user object, team data may be accessed through the team object or any team member's user object. It is therefore also possible to override team data for specific team members because the user data has a higher priority than team data.

Administrator Client

The administrator client automatically loads a new group editor tab when team support is enabled for the server. This group editor features a slightly remodelled

version of the user interface that has been used in the Santa Clara CodeBreaker experiment (see chapter 6.4.2 on page 122).

Like its predecessor, the administrator client provides operations to create and delete groups and to assign, reassign and move students to and from these groups. As an aid for the teacher, it also offers various automatic allocations; at the moment, random assignments and assignments based on a quick knowledge test are possible. Students can be promoted to team leaders with additional privileges for modifications on the group structure.

These privileges can be adjusted on a very fine level at any time. For four different groups of operations the minimum level needed for access can be determined as "student" (all users have access), "team leader" (only team leaders and above), "administrator" (only the teacher or the assistant) or "locked" (currently no access is granted):

1. *adding and removing groups* allows to create new groups and to delete unused groups,

2. *assigning and reassigning members* allows to move a user to a group or the unassigned space,

3. *inviting members* allows to invite students to become team members,

4. *leaving or joining groups* is a special category that is either "open" or "locked"; when open, students can leave or join groups as they like.

Student Client

Similar to the administrator client, the student client loads an additional screen with team information when team support is enabled. This screen incorporates all information about the current team the student is assigned to: the team name, other team members and their status.

Depending on the privileges granted to the students concerning group management, a menu is shown that allows invoking additional screens. These management screens

currently provide the means to create new teams, select students to invite (optionally based on additional information selected by the teacher), accept or decline an invitation or to remove students from a team.

7.2.2 The TeamQuiz Service

As a first experiment, we tried to find out how the quiz service of the WIL/MA toolkit can be used for team work in a lecture. A very early version with only very little support for group interaction was implemented and used during the study in the Summer semester 2004 (see chapter 6.1 on page 102): three teams played against each other in a small contest. The students in the teams received the same questions as all other students (all multiple choice), but they could see the answers of their fellow team members. As soon as an answer was marked as correct or the mark was removed, the other team members were informed by the (dis)appearance of a small red dot next to that answer.

After the course, these students were asked to share some ideas to improve the electronic group communication for this service. Some of the conceptions were quite useful and together with own ideas and experiences, a design concept for the TeamQuiz service, based on multiple choice questions, was produced.

Group communication in the TeamQuiz service is divided into two parts: global and specific consultation. Global communication allows the team members to agree on generic topics, particularly the assignment of activities, i.e., the questions each team members feels responsible to work on. The “dispatching room” can be invoked with a button at the top of the quiz sheet. It includes an activity matrix to match the team members with the exercises contained in the sheet. Furthermore, a small chat room is provided for general discussion. When something in the dispatching room changes, the button on the main quiz sheet will reflect this change with a red dot.

More important is the specific consultation bound to each answer item of the quiz questions. The idea is that all or only a selection of team members give their opinion about each item, i.e., whether they think that the answer is correct or wrong. The average of these opinions is presented in form of a coloured dot inside the check boxes

for each team member: the dot is coloured red and placed in the lower right corner when the answer is considered wrong by all team members whereas a green dot in the upper left corner means that the team members agree that the answer is right. For non-uniform votes, the dot will be coloured and placed somewhere in-between these extremes.

The students can enter their opinion by setting a mark at a certain position on a ruler on the right side of the check box. The initial value for each item and team member is "0", a white space in the middle of the ruler (neutral, the student has no opinion). From the centre to the right, the ruler changes its colour to green. Values on this section are positive and represent the opinion that the answer should be considered as correct. The farther the mark is off the centre, the more confident the student feels with his or her opinion (positive values from +1 to +3). From the centre to the left, the ruler changes the colour to red representing the opinion that the check box should be left unchecked. The confidence level determines the weight of the team member for the calculation of the mean value.

An array of red, green or grey dots on the right side of the ruler give a quick overview of the opinions of each team member: grey for undecided, green for a vote to check this item and red for a vote to consider the answer as wrong. A small blue frame around the dot indicates that the corresponding team member has also provided a small textual explanation to confirm or confine the given opinion. By clicking somewhere in this array, the detailed votes of each team member as well as the explanations are presented on an extra screen that also provides a text editor to add or change one's own statement.

The team members are not bound to the general team vote and can check or uncheck the items as they like. They are, however, asked to confirm their action if one or more of their answers deviate from a strong voting. For the team rating, the average voting is accounted for; the individual voting is solely based on the students' answers.

Some design concepts for the final version of the TeamQuiz service are provided in Figure 7.1 on the facing page.

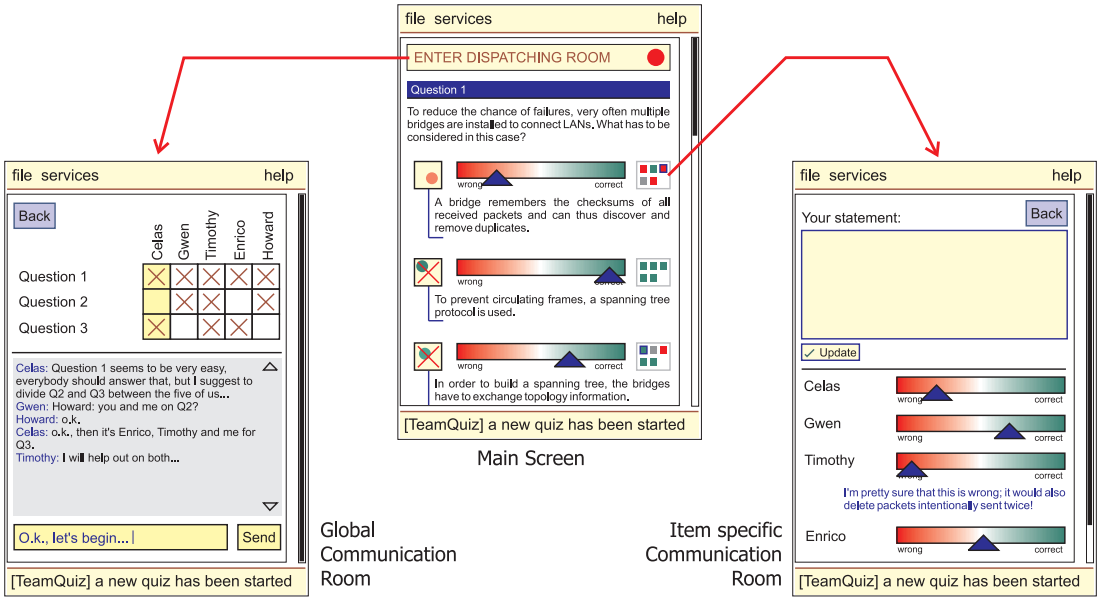

Figure 7.1: Design concepts for the TeamQuiz service

7.3 Participatory Simulations

Our second approach was to investigate a rather new educational method that, while being applicable to only very specific topics, promises to involve the students very deeply and thus to produce a good learning success. This method is called *"participatory simulation"*, a combination of a very active role play and a simulated, tightly delimited micro-cosmos.

In a simulation, a well defined part of reality is reproduced in a certain way, so that the model behaves just like the original within the given boundaries. This characteristic unfolds a wide range of possible uses:

- In a simulation, the parameters do not have to adhere to realistic defaults. While it is very hard to create extreme environmental conditions in a laboratory, it is very easy to simulate them in a model. This way, many sources of error can be eliminated by testing an assumption first in a simulation before a real-life experiment is conducted. Example: the impact of very high pressure on a new material that is to be used for deep sea probes.

- The results of a simulation do not have any impact on the real world. Failed experiments can be repeated infinitely, and dangerous experiments or experiments with a possibly disastrous outcome can be performed without concern. Example: the propagation speed of a toxic substance in ground water or the effects of a meteoric impact on the earth.

- The visualisation of the model in a simulation is not restricted to the visual light spectrum. Instead or additionally, other kinds of information can be displayed. Examples: temperature, air flow, vibration, ultra-violet light or the distribution of microscopic particles.

- For educational purposes, a complex set of principles can be reduced to a representative subset of only the most important and sustainable rules. By changing the model's parameters and analysing the results, comprehension of the modelled real world facet is promoted [KW01]. Step by step, the rule set can be expanded until the model is very close to reality. Example: In a simulation of a thrown ball, first only angle, power and the gravitational constant are considered. Later, air resistance, weight and size of the ball, wind and the effects of the ball's surface finish are added.

Simulations can be static or dynamic. In a static simulation, all necessary parameters are programmed beforehand. During the simulation, no changes are possible. In most cases, static simulations have a predetermined termination and only a certain set of values is of interest (like the first occurrence of a specific event or the final state of an object). Static simulations are generally used for extremely complex models that require a long time to compute.

Dynamic simulations are more interactive. Based on a set of parameters, the simulation runs until it is terminated by the user or a certain state is reached that does not allow continuation. The user is provided with continuously updated information and can feed the simulation at any time with new data, like inducing a new event or modifying a base parameter. User input is immediately registered and incorporated into the model. This way, the user can see the results of his or her action right away.

Participatory simulations are based on dynamic simulations with a strong educational background. Complex topics that are usually very hard to understand are

remodelled and simplified. However, while the common dynamic simulation is usually designed for a single user, participatory simulations are used by many students simultaneously. Therefore, the model itself is divided into a certain number of discrete facets which are connected by strong interdependencies. Participants are only allowed to modify the values of the facet they have been assigned to. In order to master the simulation, they have to communicate with other participants and coordinate the individual actions and by doing so develop a very deep understanding of the underlying topic [CHB01].

7.3.1 Related Projects

One of the first projects that came very close to our definition of a participatory simulation is the "Thinking Tags" experiment at the MIT. Thinking Tags were small, electronic devices with different LEDs and an infrared port. Participants carried these tags on a string around the neck. When two tag bearers met, the devices started to communicate and to exchange specified data[Bor96].

Various applications were tested. One of them was a simulation modelling the spread of a disease (the "Virus Badge Game"). The tags were programmed as "normal", "immune" or "infected" and given to a number of students. Due to a predefined incubation period, the students did not know which group they belong to. Whenever an "infected" participant came close to a "normal" student, the Thinking Tags of these two persons determined randomly whether the disease was passed. Later, the collected data was analysed in the class.

Later on, the MIT also started to create participatory simulations with PDAs; a number of these applications can be downloaded from their Webpage [Klo05]. Most of them used the infrared port of the PDAs to communicate, but they also experimented with wireless LAN or additional sensors, like GPS. In an exemplary simulation called "Environmental Detectives", the students were told to find the source of a contamination of the ground water. With the PDAs, they could perform virtual sensory measurements (the result of these was determined by the position and the corresponding entry in a database). After a two-hour-field trip, the teams had to file a final report.

All these simulations, however, are single applications without a common framework or any other kind of support for the programmers. The first endeavour to establish a substantial base for the development of interactive simulations has been made at the Northwestern University in Illinois. In the "Center for Connected Learning and Computer-Based Modelling" the software system *NetLogo* was implemented that allowed users to create new simulations with an easy-to-use interface (GUI) builder and a very intuitive programming language[1] [WHF99, TW04]. NetLogo itself was written in Java, so NetLogo simulations could be published as a Java applet on a WWW site.

While NetLogo alone did not support multi-user scenarios, the extension *HubNet* finally made it possible for people with only little programming experience to create participatory simulations [WS99]. HubNet provides the means for two different scenarios. In the first scenario, a common visualisation of the model is projected for the whole class, and the individual students use infrared-capable pocket calculators to remotely control their part of the simulation. The second scenario requires all participants to have a notebook PC to display their individual view of the model.

A famous exemplary participatory simulation that is included in the HubNet-/NetLogo-package is "GridLock" where each student controls one traffic light in a simulated grid of streets. In order to optimise the traffic flow, the students first control their traffic light manually while coordinating their efforts with the neighbours, later they use their experiences to program automated traffic light intervals.

7.3.2 Participatory Simulations with WIL/MA

Unfortunately, the development of NetLogo or HubNet was discontinued some time ago. Therefore, we examined the applicability of HubNet for more complex simulations in order to identify the need for a new framework. In a student research project, a simulation modelling a stock exchange was to be implemented using HubNet [WK04]. The simulation is supposed to give students an understanding of the

[1] The language used in NetLogo is based on LOGO, extended with agents and multithreading.

mechanisms of the stock market. A realistic market had to be simulated where students can buy and sell shares and derivatives for their virtual depot. Share prices are usually determined randomly by well-founded formulas, however, the administrator can trigger certain events with an arbitrary impact on a single stock, a group of stocks or the whole market.

Although it was indeed possible to do this with HubNet and NetLogo, we encountered many problems that complicated the development and resulted in a limited version of the originally planned scenario. Some screenshots of the final software are presented in Figure 7.2.

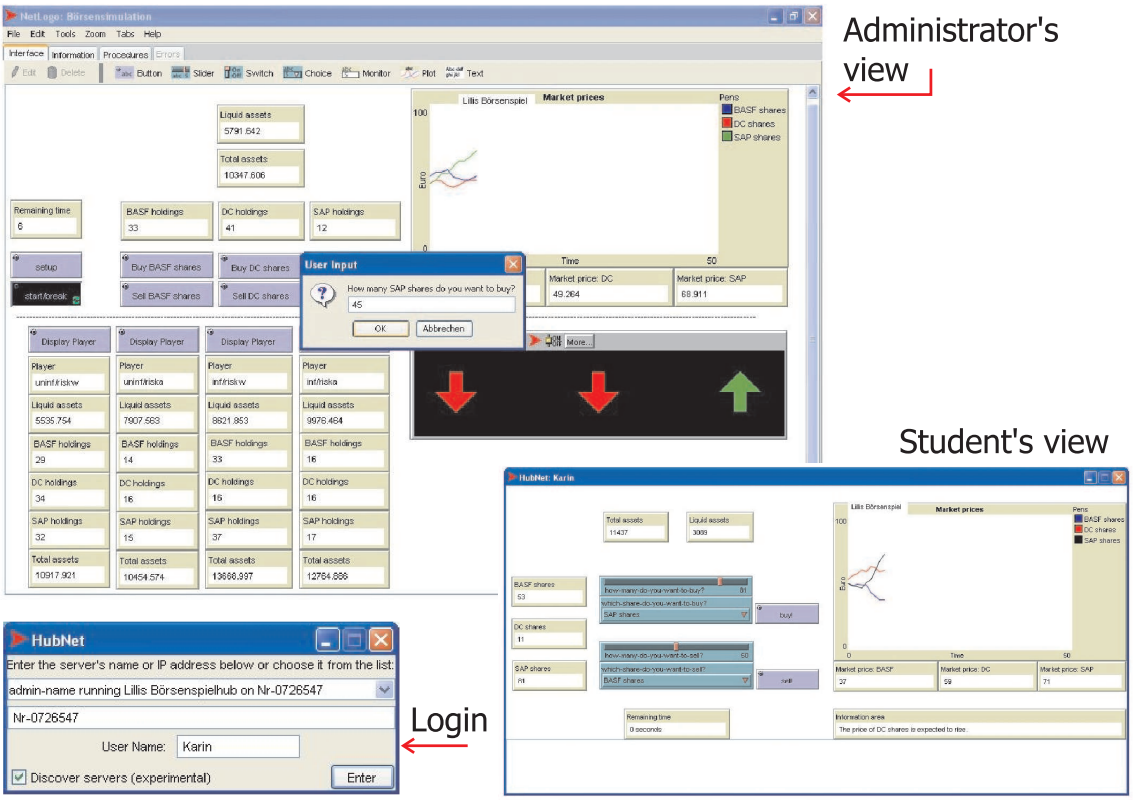

Figure 7.2: Screenshots of a stock market simulation with HubNet

The user interface builder is too limited to generate intelligent and user friendly GUIs, e.g., it was not possible to use a text field to let the user enter a number. Furthermore, the user interfaces are too static; it is not possible to dynamically change a portion of the interface based on a certain parameter. This, however, means that all elements needed for the entire simulation have to fit on a central screen. As

another inconvenience NetLogo does not allow to store any data on the client side or to send data from the client to the server. Instead, the server has to poll data from the clients which results in an endless loop of (usually unnecessary) communication attempts. These problems and further deficits in the scripting language lead to the conclusion that HubNet is a very well equipped framework for minor simulations and models with low complexity but rather unusable for more sophisticated scenarios.

Our second attempt to realise a stock market simulation was based on the WIL/MA framework. For his diploma thesis, Till Aldinger implemented a new service called "StockMarket" that could be used like any other interactive service [Ald04]. On the server side, the complicated calculations for the determination of the share prices are performed and the depots and accounts of the users are stored. The clients are provided continuously with updated information and offer a home-banking like interface to buy and sell shares or derivates, obtain information about the companies and manage the user's account.

The prominent feature of the StockMarket simulation service is the administrator. In a layout very similar to the quiz service, all tradable securities are displayed in groups: different types of securities are collated to the company they belong to, the companies are grouped to indices (e.g., DAX, STOXX, FTSE). On all three layers, preferences can be modified in detail.

On the *session (index) level*, relevant and global market data is defined: the current market tendency and a long term tendency, interest rate, oil price as an exemplary external factor and the index score. The session also defines the boundaries of the simulation, i.e., it is not possible to trade FTSE shares when the DAX session is active. Therefore, the key parameters for the simulation are bound to the session object: transaction fees, starting cash and maximum risk class.

Company related data is stored on the next level. At any time, the administrator can change the credit rating, dividend policy and a number of internal parameters of a company and thus affect the chart development. Modifications may or may not

be transparent to the students; the administrator can decide if a news flash is sent to the clients. For the elaboration, some key features of the company can be edited: number of employees, a short info text, business volume, etc; this data, however, has no direct impact on the share pricing.

On the lowest level, the administrator can edit detailed information about a *specific security*. Depending on the security type, different parameters are editable. At the moment, three different kinds of securities are supported: shares, options and bonds.

The stock market simulation is fully functional and easy to handle; a number of screenshots are included as Figure 7.3. Communication is fast and reliable and the service runs very stable. Given the high complexity of the simulation and the fact that there is still a lot of room for further expansions (like interactive charts or histories on the client devices or some more elaborated models), we educe that the WIL/MA framework is well qualified for the creation of participatory simulations.

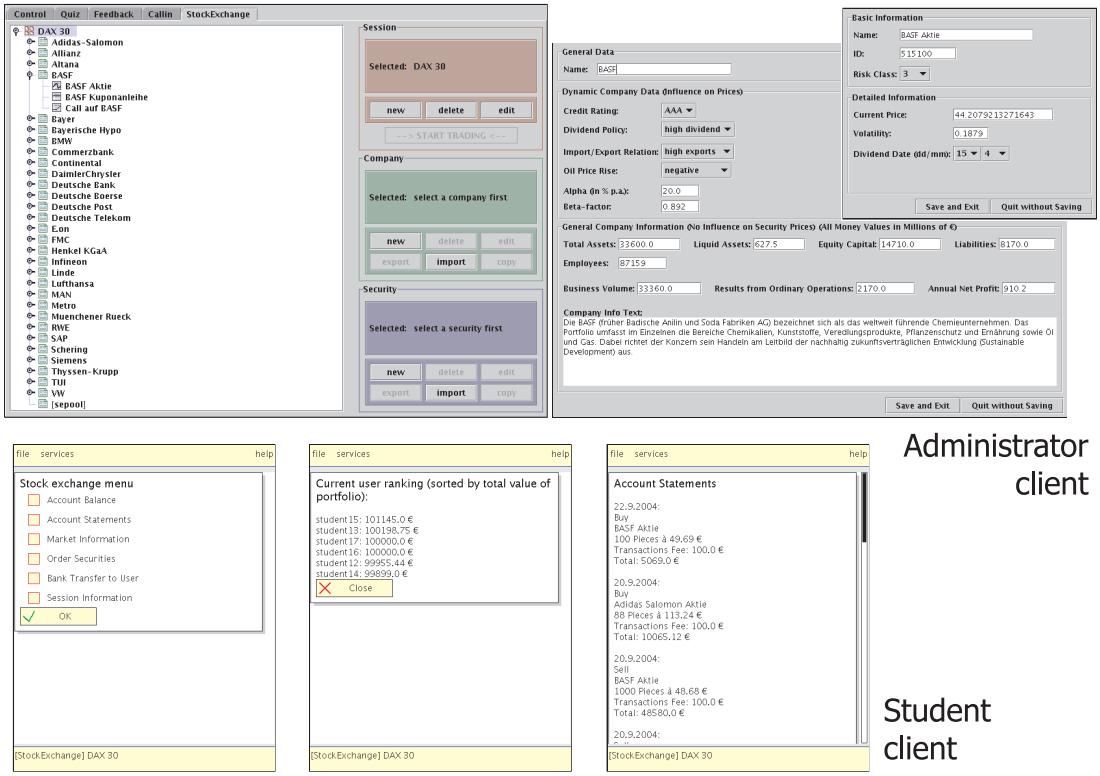

Figure 7.3: Screenshots of a stock market simulation with WIL/MA

7.3.3 The PartSim Framework

There are, however, several problems with using WIL/MA for a complex interactive simulation. First of all, the development effort is very high, particularly when compared to the NetLogo construction kit. The methods and interfaces provided by WIL/MA are highly adapted for the interactive lecture and have to be improved or partially rededicated for other uses. The most important differences between an interactive lecture and a participatory simulation are:

- In a participatory simulation, the communication structure is different from the 1-to-n-to-1 communication of the interactive lecture. Usually, the data received by the server is not addressed to a lecturer but has to be analysed, computed and integrated into a global model. Changes in the model have to be propagated back to the participants. The teacher or administrator has no important role in the scenario, except starting and pausing the simulation or changing basic parameters to increase the difficulty.

- The data input and output in an interactive lecture is much simpler than in simulations. The student clients in a lecture usually can live with only a small set of building blocks for the GUI, while a simulation generally demands highly interactive and graphically complex visualisations.

- Interactive lectures have a very strict user role model: a number of students with equal tasks and one or two teachers (or administrators). In a simulation, however, there are often a large number of different roles with different access rights.

- The running time of simulations may be much longer than that of a lecture. The stock market simulation, for example, could by used for a complete semester.

- The administrator in a simulation, being only a supporter and observer, has to be able to access portions of a generally large amount of data very quickly, but keeping a peripheral administrator client updated continuously would be very inefficient given the complex underlying model. Furthermore, there is no need for multiple administrator access nodes.

For these reasons, we decided to create a spin-off project called "PartSim", based on several core elements of WIL/MA. A first version was implemented as a diploma thesis by Giovanni Falcone [Fal04]. The architecture was thinned out noticeable: only one type of client (the student client) is supported. The administrator access is merged into the server, as is the public display of the information. On the other hand, support for different roles and groups is included in the server core [FSEA05].

The participatory simulations are packaged into modules that can be loaded with the administration window once the server has been started. Each module contains a control interface for the teacher, the user interfaces for the students and the logic of the simulation. Since simulations are much more complex than the interactive services of the WIL/MA system, only one module can be run at a time. During an initial phase after the simulation has been loaded, the students can log in with user name and password. Also any kind of information that is needed by the software beyond that is collected (e.g., GPS position, familiarity of the student with the simulation, ...). The teacher then assigns roles and - if applicable - creates teams of students. After that, the actual simulation begins. At any time, the simulation can be paused, e.g., to solve a problem or to explain something. Furthermore, it is possible to split the simulation into several distinct parts, so that it is very easy to create a curriculum with alternating theoretical phases (lecturing), practical phases (simulation) and discussions.

The students' client is divided into multiple tabs. The first tab always shows an explanation, telling the student exactly what to do during a certain phase of the simulation. Further tabs are then filled with simulation-specific information or control interfaces.

Like WIL/MA, PartSim was implemented in Java, but for the student clients a newer SDK was used. "EWE" is not a Sun-certified Java dialect, but highly compatible with Java 2 ME, and it provides virtual machines for all major platforms [Ewe05]. A major advantage of EWE, compared to PersonalJava used for WIL/MA, is that a lot of different and for the most part very complex and adaptive user interface components are offered. This and a strong support for the creation of the client user interface in the PartSim framework renders development of new simulations much easier.

For the future, it is planned to back up the development with authoring tools (as provided by NetLogo for example). However, the migration to an inferior scripting language does not make much sense; instead the strengths of Java will be backed up with an abundant set of prepared solutions.

A first example for a PartSim simulation was also developed within the scope of the diploma thesis. The simulation reproduces a network with a multitude of IP routers. Each participant controls one router and has to ensure that received packets are forwarded correctly so that they eventually reach the addressee. In a small curriculum designed for the simulation, the students first have to use the *Flooding* routing protocol in several levels of refinement, i.e., all packets first have to be forwarded to all known routers, then only to the routers the packet did not come from and in a final step a TTL parameter has to be considered. As a second phase, the routing protocol RIP [Mal94] is introduced, again with several distinct phases ,and last but not least, the students have to simulate OSPF [Moy98][Tan02]. Screenshots of this simulation are shown in Figure 7.4.

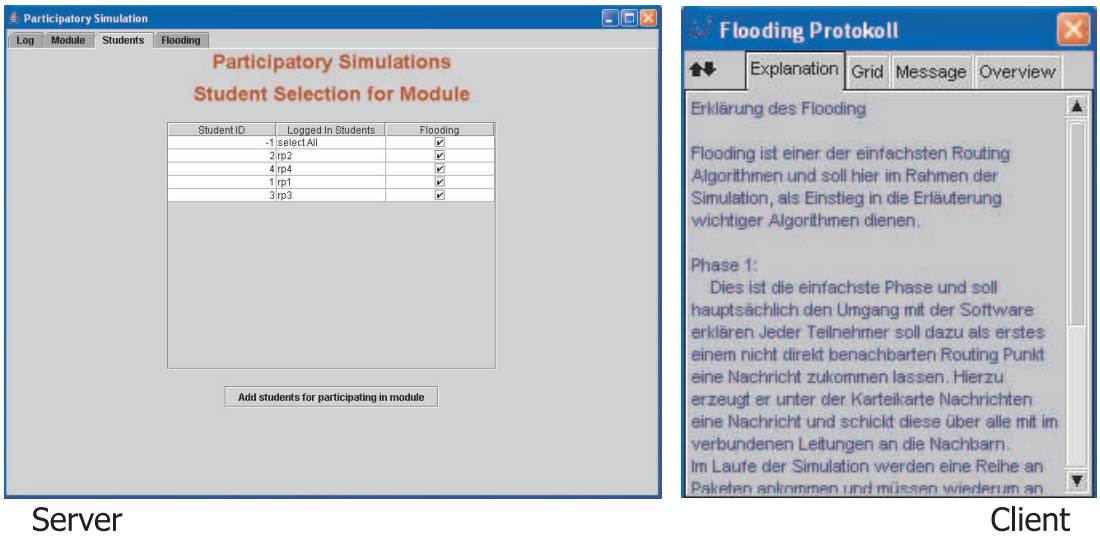

Figure 7.4: Screenshots of a routing simulation with PartSim

This simulation was then tested with about 15 students of a 12th grade computer science class in a high school in Mosbach i.Odenwald. After a short introduction, the students were equipped with PocketPCs and logged into the server. After demonstrating the Flooding protocol and the many problems associated with it, we successfully

collected ideas to improve the protocol in a group discussion. This could be repeated after the introduction of the TTL parameter: With only little additional information, one student managed to formulate the basic idea of RIP by himself. Unfortunately, we had to stop the curriculum before we could address the last part (OSPF), because we ran out of time. Pre- and post-tests and a short feedback form were used to evaluate the experiment.

The students were very interested and motivated, even though much more time was used than initially expected. Although none of them had any experience with networking before, they were able to answer the questions in the post-test very well. Furthermore, we got very promising feedback of both, the students and their teacher, who would like to repeat this experiment in the near future.

As a special feature, the simulation was position-aware. If possible, the position of the student was queried from a GPS receiver and transmitted to the server before and between the phases. Thus, the students could create the network structure simply by positioning themselves in an open space and actually see the other routers (in form of their fellow students) in reality and on their screen. We expect that the success of a participatory simulation can be improved by involving the students deeper into the model. Next to traditional means of input (keyboard, mouse, stylus), a large amount of sensory data can be used for a more profound immersion of the students, e.g., position, temperature, orientation, motion, velocity or loudness. This will be subject to future experiments.

8 Conclusion and Outlook

Lectures are a very valuable part of higher education. In terms of efficiency, there is no other educational setting that allows the transfer of knowledge to nearly as many students as the lecture does, learning with a text book being the only reasonable alternative. But still, the lecture can be seen as a very social and adaptive form of presentation. Committed lecturers can alter the contents of their lecture at any time to include up-to-date information or new research results, or they can modify or emphasise specific parts to align their course with other course offerings. A lecture is subject to constant change, this is a characteristic that a textbook most definitely does not have. Furthermore, lectures are a vital part of the social life of students. Most other educational scenarios are founded on smaller groups whereas a lecture usually pools a large part of the students of a semester.

There are, however, some disadvantages of the lecture, for which it has been criticised frequently over the last two centuries. Most important in this context are motivational deficits leading to a very low learning success. This is due to the fact that lectures usually are a monolithic kind of presentation of a single person for a rather long time, whereas the students have no or only little room to interact. Also feedback is missing for both, the teacher and the students: the teacher never knows if the students did indeed understand the topic or how they feel about the lecture, and the students have no chance to test their newly gained knowledge.

Using modern technology, our aim was to improve interactivity and motivation of the students in a lecture. Therefore, we implemented a set of tools called "WIL/MA" with which interactive services could be established during the lecture. The students could access these services with small and light-weight PocketPCs or notebooks they were equipped with during the lecture. Using wireless LAN, no time consuming or

costly installation is needed; all but a few required parts use radio to communicate and are equipped with batteries.

For the architecture we envisioned a strict separation of tasks, combined with a modular organisation to ease the development of extensions. The management of the basic communication framework is included in the server part of the software; the logic for the interactive services is contained in individual modules which are loaded by the server in the start up phase. Clients are "intelligent" terminals which do not contain any service logic but are able to do some calculations and decisions on their own as defined in plug-in modules. Storage of persistent service data or logging of events is done solely by the administrator client; the server only keeps transient data.

Beginning with a prototype, we developed a complete open source software system with easy to use APIs for the development of new services. The setup of WIL/MA in a lecture hall is supported by a number of tools, allowing an installation in less than 10 minutes. With a complex, reliable and very efficient communication system, all required communication patterns arising in an interactive lecture are offered. Data storage is performed with platform independent XML files, thus no database backup is needed. Various stress tests have shown that the software in combination with inexpensive off-the-shelf hardware is capable of handling well over 120 students. The final release featured two different clients tailored to the requirements of the students and the administrator. Both use a unified plug-in model allowing to use all services with one piece of software.

In five elaborate field studies we have shown that the interactive lecture indeed meets most of the desired effects. The acceptance of the students for this new teaching scenario is very high, and the improved interactivity (particularly by means of the quiz service) achieves a higher motivation and in the end a higher learning success. We could also show that the tools are generally considered user friendly and appropriate for the task by both the students of a technical and a liberal arts department. During the experiments, we could also define a set of guidelines for the teacher to use the interactive services as efficiently as possible.

Furthermore, in cooperation with the Stanford Center for Innovations in Learning,

we conducted a large-scale experiment in a middle school in Santa Clara. In this experiment, a specially prepared version of the WIL/MA tools was used to run an elaborated four-week math course with two classes of about 30 children to investigate if the interactive lecture scenario can be applied to K-12. First results indeed show that motivation and learning success of the students was higher than those of the students in classes with pen- & paper-based solutions.

After the first public release of the WIL/MA toolkit, we started to investigate new ways to improve the students' experience in a classroom. This led to a first prototype of team support in quizzes that will be improved and extended to other services in the future. Beyond that, we extended WIL/MA to participatory simulations as a new educational setting. For that, we evaluated the capabilities of HubNet and WIL/MA for the development of team simulations and decided to found a spin-off project incorporating the advantages of WIL/MA with a better adaptation for the requirements of participatory simulations.

Participatory simulations and interactive group support will be investigated in the future, but the studies in the last two semesters and a number of surveys also revealed much expandability in the WIL/MA framework. In our efforts, we concentrated on three rather basic services: quiz, feedback and call-in; the task now is to invent and implement new and innovative services for the interactive lecture. But also the existing services can be improved. The quiz service, for example, can be enriched with new question types which can be used to create more complex and more demanding exercises for the students.

Last but not least, our recent efforts to publicise WIL/MA and the interactive lecture finally yield fruit, but still a lot of work has to be done to make WIL/MA a widely used application in lectures of modern universities.

Bibliography

[AAB+98] G.D. Abowd, A.C. Atkeson, J. Brotherton, T. Enqvist, P. Gulley, and J. LeMon. **Investigating the capture, integration and access problem of ubiquitous computing in an educational setting**. In *Proceedings of the Conference On Human Factors in Computing Systems (SIGCHI'98)*, Los Angeles, CA, U.S.A., 1998.

[AAF+96] G.D. Abowd, C.G. Atkeson, A. Feinstein, C. Hmelo, R. Kooper, S. Long, N. Sawhney, and M. Tani. **Teaching and Learning as Multimedia Authoring: The Classroom 2000 Project**. In *In proceedings of ACM Multimedia '96*, pages 187–198, Boston, U.S.A., 1996.

[AAV+03] R.J. Anderson, R. Anderson, T. VanDeGrift, S.A. Wolfman, and K. Yasuhara. **Promoting Interaction in Large Classes with Computer-Mediated Feedback**. In *Proceedings of the International Conference on Computer Supported Collaborative Learning (CSCL'03)*, pages 119–123, Bergen, Norway, Jun. 2003.

[Ald04] T. Aldinger. **Entwicklung einer verteilten Börsensimulation**. Master's thesis, University of Mannheim, Faculty of Computer Science, Oct. 2004. (German).

[Ape99] H.J. Apel. **Die Vorlesung: Einführung in eine akademische Lehrform**. Böhlau, Cologne, Germany, 1999. (German).

[Bar32] F.C. Bartlett. **Remembering: A study in experimental and social psychology**. Cambridge University Press, London, U.K., 1932.

[BBC00] J.D. Bransford, A.L. Brown, and R.R. Cocking. **How People Learn. Brain, Mind, Experience and School**. National Academic Press, Washington D.C., U.S.A., 2000.

[Ber68] D.C. Berliner. **The effects of test-like events and note-taking on learning from lecture instruction**. Unpublished doctoral dissertation, Stanford University; taken from Gage, Berliner: Educational Psychology (1975), 1968.

[BGW01] N. Borisov, I. Goldberg, and D. Wagner. **Intercepting Mobile Communications: The Insecurity of 802.11**. In *Proceedings of the Seventh Annual International Conference on Mobile Computing And Networking (ACM MOBICOM 2001)*, pages 180–189, Rome, Jul. 2001.

[BJP76] D.A. Bligh, D. Jaques, and D.W. Piper. **Seven Decisions when Teaching Students**. Intellect Books, 1976.

[Bli00] D.A. Bligh. **What's the use of lectures?** Jossey-Bass Publishers, San Francisco, 2000.

[Blo53] B.S. Bloom. **Thought processes in lectures and discussions**. *Journal of General Education*, No. 7, pages 160–169, 1953.

[Bor96] R. Borovoy. **Things that blink: Computationally augmented name tags**. *IBM Systems Journal*, Vol. 35(No. 3 & 4), pages 488–495, 1996.

[BP02] Andrea Butter and David Pogue. **Piloting Palm: The Inside Story of Palm, Handspring and the Birth of the Billion Dollar Handheld Industry**. Wiley, 1st edition, Feb. 2002.

[BS05] Bluetooth SIG, **http://www.bluetooth.org/**. WWW, last visited: Jan. 2005.

[Car01] L. Carson. **Teaching Power**. In H. Edwards, B. Smith, and G. Webb, editors, *Lecturing. Case studies, experience and practice*. Kogan Page Limited, London, U.K., 2001.

[CHB01] G.K.W.K. Chung, T.C. Harmon, and E.L. Baker. **The impact of a simulation-based learning design project on student learning**. *IEEE Transactions on Education*, Vol. 44(No. 4), pages 390–398, Nov. 2001.

[CMD00] F. Chen, B. Myers, and D.Yaron. **Using Handheld Devices for Tests in Classes**. Technical Report CMU-CS-00-152 and CMU-HCII-00-101, Carnegon Mellon Universtiy, School of Computer Science and Human Computer Interaction Institute, Jul. 2000.

[Cro77] L. Cronbach. **Aptitudes and Instructional Methods: A Handbook for Research on Interactions**. Irvington Publishers, New York, 1977.

[DDFW03a] P. Dawabi, L. Dietz, A. Fernandez, and M. Wessner. **ConcertStudeo: Using PDAs to support face-to-face learning**. In *Proceedings of the International Conference on Computer Supported Collaborative Learning (CSCL'03)*, pages 235–237, Bergen, Norway, Jun. 2003.

[DDFW03b] L. Dietz, P. Dawabi, A. Ferndandez, and M. Wessner. **Improving Face-to-Face Learning with ConcertStudeo**. *Learning Technology, IEEE Computer Society*, Vol. 5(No. 2), pages 51–57, 2003.

[DGL+96] R.J. Dufresne, W.J. Gerace, W.J. Leonard, J.P. Mestre, and L. Wenk. **Classtalk: A Classroom Communication System for Active Learning**. *Journal of Computing in Higher Education*, No. 7, pages 3–47, 1996.

[DLB+98] R.C. Davis, J. Lin, J.A. Brotherton, J.A. Landay, M.N. Price, and B.N. Schilit. **A Framework for Sharing Handwritten Notes**. In *Proceedings of the Annual ACM Symposium on User Interface Software and Technology (UIST'98)*, pages 119–120, San Francisco, U.S.A., 1998.

[DLC+99] R.C. Davis, J.A. Landay, V. Chen, J. Huang, R.B. Lee, F.C. Li, J. Lin, C.B. Morrey, B. Schleimer, M.N. Price, and B.N. Schilit. **Notepals: Lightweight Note Sharing by the Group, for the Group**. In *Proceedings of the Conference On Human Factors in Computing Systems (SIGCHI'99)*, pages 338–345, May 1999.

[Dro93] R. Droms. **Dynamic Host Configuration Protocol**. RFC 1541, IETF, Oct. 1993.

[Dub95] R. Dubs. **Lehrerverhalten. Ein Beitrag zur Interaktion von Lehrenden und Lernenden im Unterricht**. Verlag SKV, Zürich, Switzerland, 1995. (German).

[Dub00] Rolf Dubs. **Universitätsstudium - Anforderungen aus der Sicht der Lehr- und Lernforschung**, volume 8. Universitäre Hochschule Luzern, Luzern, Switzerland, 2000. (German).

[eIN05] eInstruction, **http://www.einstruction.com/**. WWW, last visited: Jan. 2005.

[Ent81] N. Entwhistle. **Styles of Learning and Teaching: An Integrated Outline of Educational Psychology for Students, Teachers and Lectures**. John Wiley & Sons, Edinburgh, U.K., 1981.

[Ern95] P. Ernest. **The one and the many**. In L. Steffe and J. Gale, editors, *Constructivism in education*, pages 459–486. Erlbaum, Hillsdale, NJ, U.S.A., 1995.

[Ewe05] EWE Soft, **http://www.ewesoft.org/**. WWW, last visited: Jan. 2005.

[Fal04] G. Falcone. **Positionsbasierte partizipierende Simulationen**. Master's thesis, University of Mannheim, Faculty of Computer Science, Aug. 2004. (German).

[FJM98] Dave W. Farthing, Dave M. Jones, and Duncan McPhee. **Permutational multiple-choice questions: an objective and efficient alternative to essay-type examination questions**. In *Proceedings of the 6th annual conference on the teaching of computing and the 3rd annual conference on Integrating technology into computer science education*, pages 81–85, 1998.

[FSEA05] G. Falcone, N. Scheele, W. Effelsberg, and C. Atkinson. **PartSim - Ein System zur Unterstützung interaktiver Simulationen in der Lehre**. In *Proceedings of "die 3. eLearning Fachtagung der Gesellschaft für Informatik" (DeLFI)*, Rostock, Germany, (accepted) Sep. 2005. (German).

[Gag65] R.M. Gagne. **The conditions of Learning**. Holt, Rinehart & Winston, 1965.

[GB96a] N.L. Gage and D.C. Berliner. **Educational Psychology**. Houghton, Mifflin, Boston, Massachussetts, 6th edition, 1996.

[GB96b] N.L. Gage and D.C. Berliner. **Pädagogische Psychologie**. PVU, Weinheim, Germany, 5th edition, 1996. (German).

[GBL99] S. Greenberg, M. Boyle, and J. LaBarge. **PDAs and Shared Public Displays: Making Personal Information Public, and Public Information Personal**. *Personal Technologies, Elsevier*, Vol. 3(No. 1), Mar. 1999.

[Gib81] G. Gibbs. **Twenty terrible reasons for lecturing**. *SCED Occasional Paper*, No. 8, 1981.

[GNU05] GNU, **http://www.gnu.org/copyleft/gpl.html**. WWW, last visited: Jan. 2005.

[Gol01] S. Goldman. **Technology in the Mathematics Classroom: Guidelines from the Field**. *ERIC Update. ERIC Clearinghouse on Information and Technology*, Vol. 22(No. 2), 2001.

[GPM04a] S. Goldman, R. Pea, and H. Maldonado. **Emerging social engineering in the wireless classroom**. In *In Proceedings of the International Conference on the Learning Sciences*, Santa Monica, CA, U.S.A., Jun. 2004.

[GPM+04b] S. Goldman, R. Pea, H. Maldonado, L. Martin, and T. White. **Functioning in the Wireless Classroom**. In *Proceeding of the 2nd IEEE International Workshop on Wireless and Mobile Technologies in Education (WMTE 2004)*, pages 75–82, Taoyuan, Taiwan, Mar. 2004.

[Hae69] H. Haecker. **Kritische Betrachtung zur Vorlesung**. *Didactica*, No. 4, pages 261–271, 1969. (German).

[Hil01] F. Hillebrand. **GSM & UMTS: The Creation of Global Mobile Communications**. John Wiley & Sons, Dec. 2001.

[Hon96] P. Honebein. **Seven goals for the design of constructivist learning environments**. In B. Wilson, editor, *Constructivist learning environments*, pages 17–24. Educational Technology Publications, New Jersey, U.S.A., 1996.

[HSS85] H. Heckhausen, H.D. Schmalt, and K. Schneider. **Achievement motivation in perspective**. Academic Press, Orlando, FL, U.S.A., 1985.

[IEE99a] IEEE. **Supplement to 802.11, Wireless LAN MAC and PHY specifications: Higher speed Physical Layer (PHY) extension in the 2.4 GHz band**. IEEE Standard 802.11b, IEEE, 1999.

[IEE99b] IEEE. **Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications**. IEEE Standard 802.11, IEEE, 1999.

[IEE03] IEEE. **Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) - Amendment 4: Further Higher-Speed Physical Layer Extension in the 2.4 GHz band**. IEEE Standard 802.11g, IEEE, 2003.

[Jon23] H.E. Jones. **Experimental studies of college teaching**. *Archives of Psychology*, Vol. 68, 1923.

[Jon94] D. Jonassen. **Thinking Technology**. *Journal of Educational Technology*, No. 34, pages 34–37, 1994.

[KB04] C.D. Knutson and J.M. Brown. **IrDA Principles and Protocols: The IrDA Library**, volume 1. MCL Press, May 2004.

[KH02] M. Kroll and S. Haustein. **Java 2 Micro Edition (J2ME) Application Development**. Pearson Education, 1st edition, Jun 2002.

[KK88] J.A. Kulik and C. Kulik. **Timing of Feedback and Verbal Learning**. *Review of Educational Research*, Vol. 58, pages 79–97, 1988.

[Klo05] E. Kloepfer, **http://education.mit.edu/ar/**. WWW, last visited: Jan. 2005.

[KPCB02] V.M. Kern, J.M. Pernigotti, M.M. Calegaro, and M. Bento. **Peer review in engineering education: speeding up learning, looking for a paradigm shift**. In *Proceedings of the seventh International Conference on Engineering and Technology Education (INTERTECH)*, Santos, Greece, 2002.

[KW01] A. Krapp and B. Weidenmann, editors. **Pädagogische Psychologie**. Beltz Psychologie Verlags Union, Weinheim, Germany, 2001. (German).

[LE04] H.C. Liebig and W. Effelsberg. **Computer-supported Formation of Virtual Learning Groups based on Proficiency Levels**. In *Proceedings of the World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA'04)*, Lugano, Switzerland, Jun. 2004.

[LL05] LectureLab Team, **http://www.lecturelab.de/**. WWW, last visited: Jan. 2005.

[Llo68] D.M. Lloyd. **A concept of improvement of learning response in the taught lesson**. *Visual Education*, pages 23–25, Oct. 1968.

[LM96] J. Lompscher and H. Mandl, editors. **Lehr- und Lernprobleme im Studium: Bedingungen und Veränderungsmöglichkeiten**. Verlag Hans Huber, Bern, Switzerland, 1996. (German).

[LOB05] Legion of the Bouncy Castle, **http://www.bouncycastle.org/**. WWW, last visited: Jan. 2005.

[Mal94] G. Malkin. **RIP Version 2 Protocol Analysis**. RFC 1387, IETF, Nov. 1994.

[MB01] B.A. Miller and C. Bisdikian. **Bluetooth Revealed: The Insider's Guide to an Open Specification for Global Wireless Communication**. Prentice Hall PTR, 2nd edition, Dec. 2001.

[McK68] W.J. McKeachie. **Research in Teaching: The Gap Between Theory and Practice**. In C.B.T. Lee, editor, *Improving Council on Education*. American Council on Education, Washington D.C., U.S.A., 1968.

[McL70] J. McLeish. **Students' attitudes and college environments**. Cambridge Institute of Education, 1970.

[McL76] J. McLeish. **The Lecture Methods**. In N.L. Gage, editor, *The psychology of teaching methods*, pages 252–301. The University of Chicago Press, Chicago, IL, U.S.A., 1976.

[MCTR94] T. Mayes, L. Coventry, A. Thomson, and R.Mason. **Learning through Telematics: A Learning Framework for Telecommunication Applications in Higher Education**. Technical report, British Telecom, Martlesham Heath, 1994.

[Mer91] M.D. Merrill. **Constructivism and instructional design**. *Educational Technology*, Vol. 3, pages 45–53, 1991.

[MK75] W.J. McKeachie and J.A. Kulik. **Effective College Teaching**. In F.N. Kerlinger, editor, *Review on research in education*, volume 3. American Educational Research Association, Washinton D.C., U.S.A., 1975.

[ML03] M. McCabe and I. Lucas. **Engagement with Mathematics in an Interactive Classroom**. In *Proceedings of the 6th International Conference on Technology in Mathematics Teaching (ICTMT6)*, Volos, Greece, Oct. 2003.

[MLMI66] W.J. McKeachie, Y.G. Lin, J. Milholland, and R. Isaacson. **Student affliction motives, teacher warmth, and academic achievement**. *Journal of Personality and Scial Psychology*, Vol. 4(No. 4), pages 457–461, 1966.

[MMP73] J. McLeish, W. Matheson, and J. Park. **The psychology of the learning group**. Hutchinson & Co, London, U.K., 1973.

[Moy98] J. Moy. **OSPF Version 2**. RFC 2328, IETF, Apr. 1998.

[MSG98] B.A. Myers, H. Stiel, and R. Gargiulo. **Collaboration Using Multiple PDAs Connected to a PC**. In *Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work (CSCW'98)*, pages 285–294, Seattle, U.S.A., 1998.

[Mue00] Chris Muench. **The Windows CE Technology Tutorial: Windows Powered Solutions for the Developer**. Addison-Wesley, 1st edition, May 2000.

[Mye01] B.A. Myers. **Using Hand-Held Devices and PCs Together**. *Communications of the ACM*, Vol. 44(No. 11), pages 34–41, Nov. 2001.

[Obe05] Virtuelle Hochschule Oberrhein, **http://www.viror.de/**. WWW, last visited: Jan. 2005.

[OP99] B. O'Hara and A. Petrick. **The IEEE 802.11 Handbook: A Designer's Companion**. IEEE, 1999.

[Pau66] F. Paulsen. **Die deutschen Universitäten und das Universitätsstudium**. Georg Olms, Hildesheim, Germany, 1966. (German).

[Pet79] P.L. Peterson. **Direct instruction reconsidered**. In P.L. Peterson and H.J. Walberg, editors, *Research on Teaching Concepts, Findings and Implications*, pages 57–69. McCutchan, Berkeley, CA, U.S.A., 1979.

[Pos81] J.B. Postel, Editor. **Transmission Control Protocol**. RFC 793, IETF, Sep. 1981.

[Pos82] J.B. Postel. **Simple Mail Transfer Protocol**. RFC 821, IETF, Aug. 1982.

[Ram92] P. Ramsden. **Learning to Teach in Higher Education**. Routledge, London, 1992.

[Rei99] C.M. Reigeluth. **Instructional - Design Theories and Models: A new Paradigm of Instructional Theory**, volume 2. Lawrence Erlbaum Associates, London, U.K., 1999.

[Rek97] J. Rekimoto. **Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments**. In *Proceedings of the Annual ACM Symposium on User Interface Software and Technology (UIST'97)*, pages 31–39, Banff, Canada, 1997.

[Rek98] J. Rekimoto. **A Multiple Device Approach for Supporting Whiteboard-based Interactions**. In *Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work (CSCW'98)*, pages 344–351, Apr. 1998.

[Rhe00] F. Rheinberg. **Motivation**, volume 6. Kohlhammer, 2000. (German).

[RP02] J. Roschelle and R. Pea. **A walk on the WILD side: How wireless handhelds may change CSCL**. *International Journal of Cognition and Technology*, No. 1, pages 145–168, 2002.

[RR83] W. Rieck and U.P. Ritter. **Lernsituationen in der Hochschulausbildung**. In L. Huber, editor, *Ausbildung und Sozialisation in der Hochschule*, volume 10 of *Enzyklopädie Erziehungswissenschaften*, pages 367–400. Klett, 1983. (German).

[Sas92] E.J. Sass. **Motivation in the College Classroom: What Students Tell Us**. *Teaching of Psychology*, No. 16, pages 86–88, 1992.

[Sax05] Learning Lab Lower Saxony, **http://www.learninglab.de/**. WWW, last visited: Jan. 2005.

[Sch70] J.R. Schoen. **Use of Consciousness Sampling to Study Teaching Methods**. *Journal of Educational Research*, Vol. 63, pages 387–390, May 1970.

[Sch96] B. Schneier. **Applied Cryptography**. John Wiley & Sons, 2nd edition, 1996.

[Sch02] H. Schildt. **Java 2: The Complete Reference**. McGraw-Hill Osborne Media, 5th edition, 2002.

[Sie00] T. Siep. **An IEEE Guide: How to Find What You Need in Bluetooth Spec**. IEEE, 2000.

[Sla80] R. E. Slavin. **Cooperative Learning**. *Review of Educational Research*, No. 50, pages 313–342, 1980.

[SME+02] N. Scheele, M. Mauve, W. Effelsberg, A. Wessels, and S. Fries. **The Interactive Lecture**. Technical Report TR-02-006, Department for Mathematics and Computer Science, Jan. 2002.

[SME+03] N. Scheele, M. Mauve, W. Effelsberg, A. Wessels, H. Horz, and S. Fries. **The Interactive Lecture - A new Teaching Paradigm Based on Ubiquitous Computing**. In *Poster Proceedings of the International Conference on Computer Supported Collaborative Learning (CSCL'03)*, pages 135–137, Bergen, Norway, Jun. 2003.

[Smi01] B. Smith. **Just give us the right answer**. In H. Edwards, B. Smith, and G. Webb, editors, *Lecturing. Case studies, experience and practice*, pages 123–129. Kogan, London, U.K., 2001.

[SSC+63] K. Siegel, L.C. Siegel, R. Capretta, R.L. Jones, and H. Berkovitz. **Students thoughts during class: A criterion for educational research**. *Journal of Educational Psychology*, No. 54, pages 45–51, 1963.

[SSEW04] N. Scheele, C. Seitz, W. Effelsberg, and A. Wessels. **Mobile Devices in Interactive Lectures**. In *Proceedings of the World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA'04)*, Lugano, Switzerland, Jun. 2004.

[Sun05] SUN Inc., **http://java.sun.com/**. WWW, last visited: Jan. 2005.

[SWE04] N. Scheele, A. Wessels, and W. Effelsberg. **Die Interaktive Vorlesung in der Praxis**. In *Proceeding of "die 2. eLearning Fachtagung der Gesellschaft für Informatik" (DeLFI)*, pages 72–80, Paderborn, Germany, Sep. 2004. (German).

[Tan02] A. S. Tanenbaum. **Computer Networks**. Prentice Hall PTR, 4th edition, Sep. 2002.

[Tre51] J.M. Trenaman. **The Lenght of a Talk**. from McLeish: The Psychology of Teaching Methods (1976), 1951.

[TW04] S. Tisue and U. Wilensky. **NetLogo: Design and Implementation of a Multi-Agent Modeling Environment**. In *Proceedings of Agent '04*, Chicago, IL, U.S.A., Oct. 2004.

[USG96] R: Ulrich, K.H. Stapf, and M. Giray. **Faktoren und Prozesse des Einprägens und Erinnerns**. In D. Albert and K.H. Stapf, editors, *Gedächtnis*, Enzykloaedie der Psychologie, Themenbereich C, Theorie und Forschung, Serie II, Kognition, Band 4, pages 95–179. Hogrefe, Göttingen, Germany, 1996. (German).

[van80] R. vanHouten. **Learning through feedback: A systematic approach for improving academic performance**. Human Sciences Press, New York, U.S.A., 1980.

[VBB03] V. Vitsas, P. Barker, and A.C. Boucouvalas. **IrDA Infrared WIreless Communications: Protocol Throughput Optimization**. *IEEE WIreless Communications Magazine: Special Issue on Optical Wireless Communications*, Vol. 10(No. 2), pages 22–29, Apr. 2003.

[VM01] J. Vogel and M. Mauve. **Consistency Control for Distributed Interactive Media**. In *Proceedings of the 9th International ACM Multimedia Conference*, pages 101–109, Ottawa, Sep. 2001.

[War56] J.N. Ward. **Group-study versus lecture-demonstation method in physical science instruction for general education college students**. *Journal of Experimental Education*, Vol. 24, pages 197–210, March 1956.

[WC91] B. Wilson and P. Cole. **A review of cognitive teaching models**. *Educational Technology Reasearch and Development*, No. 39, pages 47–64, 1991.

[Wei98] F.E. Weinert. **Neue Unterrichtskonzepte zwischen gesellschaftlichen Notwendigkeiten, pädagogischen Visionen und psychologischen Möglichkeiten**. In *Proceedings of Wissen und Werte für die Welt von morgen (Bildungskongress des Bayerischen Staatsministeriums für Unterricht, Kultus, Wissenschaft und Kunst)*, pages 101–125, Munich, Germany, 1998.

[Wei99] F.E. Weinert. **Demands on education today**. *Education*, No. 60, pages 7–13, 1999.

[WFU05] Wake Forest University, **http://classinhand.wfu.edu/**. WWW, last visited: Jan. 2005.

[WH01] J.Y. Wilson and A. Havewala. **Building Powerful Platforms with Windows CE**. Addison-Wesley, 1st edition, Mar. 2001.

[WHF99] U. Wilensky, E. Hazzard, and R. Froemke. **An Extensible Modeling Toolkit for Exploring Statistical Mechanics**. In *In Proceedings of the Seventh European Logo Conference (EUROLOGO'99)*, Sofia, Bulgaria, 1999.

[WK04] Lilli Winschel and Stephan Kopf. **Entwicklung einer Börsensimulation der multiagentenbasierten Entwicklungsumgebung NetLogo**. Technical Report TR-04-007, University of Mannheim, Faculty of Computer Science, Mannheim, Germany, Oct. 2004. (German).

[WS99] U. Wilensky and W. Stroup. **Learning through Participatory Simulations: Network-Based Design for Systems Learning in Classrooms**. In *Proceedings of the International Conference on Computer Supported Collaborative Learning (CSCL'99)*, Stanford University, CA, U.S.A., Dec. 1999.

[WS05] A. Wessels and N. Scheele. **Experiences with Interactive Lectures - Considerations from the Perspective of Educational Psychology and Computer Science**. In *International Conference on Computer Supported Collaborative Learning (CSCL'05) (accepted)*, Taipeh, China, Jun. 2005.