OTTO-VON-GUERICKE-UNIVERSITÄT MAGDEBURG

FAKULTÄT FÜR INFORMATIK

Document-Oriented E-Learning Components

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von Michael Piotrowski, M.A.
geb. am 5. Juli 1972 in Aachen

Gutachter:
Prof. Dr. Dietmar Rösner
Prof. Dr. Anne Brüggemann-Klein
Prof. Dr. Helmut Schauer

Promotionskolloquium:
Magdeburg, den 17. April 2009

Document-Oriented
E-Learning Components

Design by Michael Piotrowski. Typeset with LaTeX in Bitstream Zapf Elliptical 711, Linotype Basic Commercial, and Bitstream Prestige 12 Pitch.

Abstract

This dissertation questions the common assumption that e-learning requires a *learning management system* (LMS) such as Moodle or Blackboard. Based on an analysis of the current state of the art in LMSs, we come to the conclusion that the functionality of conventional e-learning platforms consists of basic content management and communications facilities (such as forums, chats, wikis, etc.) and functionality for assessment (such as quizzes). However, only assessment functionality is actually specific to e-learning. Furthermore, the content management and communication functionality in e-learning platforms is typically restricted and often inferior when compared with the more general implementations available in Web content management systems.

Since content management systems (CMS) offer more general and more robust functions for managing content, we argue that e-learning platforms should be based on content management systems. Only assessment functions are actually specific to e-learning and need to be added to a CMS; this requires the architecture of the CMS to be modular.

As a proof of concept, we have designed and implemented the eduComponents, a component-based e-learning system architecture, realized as software components extending a general-purpose content management system with facilities for course management and assessment.

The eduComponents have been successfully used since several semesters at Otto von Guericke University and other institutions. The experience with the eduComponents gives practical evidence for the theses we have put forward in this dissertation and of the feasibility of the eduComponents approach.

The research done for this dissertation has also resulted in practical definitions for *e-learning* and *e-learning platform*, terms which are notoriously ill-defined. Based on these definitions, we have developed an innovative way to assess and to visualize the areas of functionality of e-learning environments.

Zusammenfassung

Diese Dissertation stellt die gängige Annahme in Frage, dass ein *Learning Management System* (LMS) wie Moodle oder Blackboard eine Voraussetzung für E-Learning ist. Gestützt auf eine Analyse des aktuellen Stands der Technik bei LMS kommen wir zu dem Schluss, dass die Funktionalität konventioneller Lernplattformen zum einen aus grundlegenen Content-Management- und Kommunikationsfunktionen (Foren, Chats, Wikis usw.) und zum anderen aus Funktionen für Leistungsüberprüfungen (z. B. Tests) besteht; lediglich letztere sind jedoch tatsächlich e-learning-spezifisch. Darüber hinaus ist die Content-Management- und Kommunikationsfunktionalität von E-Learning-Plattformen häufig eingeschränkt und im Vergleich zu den allgemeineren Implementationen in Content-Management-Systemen (CMS) oft schwach.

Da CMS allgemeinere und ausgereiftere Funktionen für die Verwaltung von Inhalten bieten, fordern wir, dass E-Learning-Plattformen auf CMS aufbauen sollten. Lediglich die Funktionalität für Leistungsüberprüfungen ist e-learning-spezifisch und muss zum CMS hinzugefügt werden; dies erfordert eine modulare CMS-Architektur.

Als Machbarkeitsnachweis haben wir die eduComponents entworfen und implementiert, eine komponentenbasierte E-Learning-Systemarchitektur, die in der Form von Softwarekomponenten für ein allgemeines CMS realisiert wurde und das CMS um Funktionalität für Kursverwaltung und Leistungsüberprüfung erweitert.

Die eduComponents sind seit mehreren Semestern an der Otto-von-Guericke-Universität und anderen Institutionen erfolgreich im Einsatz. Die Erfahrungen mit den eduComponents sind der praktische Nachweis für die in dieser Dissertation aufgestellten Thesen und die Tauglichkeit des eduComponents-Ansatzes.

In dieser Dissertation stellen wir außerdem praktische Definitionen für die notorisch unklaren Begriffe *E-Learning* und *E-Learning-Plattform* auf. Auf der Basis dieser Definitionen haben wir einen neuen Ansatz zur Einschätzung und Visualisierung der Funktionalität von E-Learning-Umgebungen entwickelt.

Contents

List of Figures

List of Tables

Acknowledgments

The research described in this dissertation was performed in the WDOK research group during my employment at Otto von Guericke University Magdeburg. I would like to thank Prof. Dr. Dietmar Rösner, head of WDOK and doctoral mentor, for his support and his enthusiasm for the eduComponents.

I would like to thank Dipl.-Wirtsch.-Inf. Mario Amelung, my colleague at WDOK, for the excellent, enjoyable, and effective collaboration. It was a pleasure to work with him on the eduComponents. We spent many hours discussing the design and jointly developed key parts using pair programming. We collaborated on almost all of the publications on the eduComponents, often in "extreme writing" sessions. I would like to thank Dipl.-Inf. Wolfram Fenske, then my student assistant, for his outstanding programming work and the long conversations about fine points of the implementation, UNIX, Lisp, and XEmacs.

Most of the actual writing of the dissertation was done while I worked at the Institute of Computational Linguistics at the University of Zurich. I would like to thank Prof. Dr. Michael Hess, head of the institute, for being able to reconcile my project work with the work on my dissertation. I would like to thank my Zurich colleagues for creating a pleasant working environment, and in particular Martin Warin, lic. phil. Maya Bangerter, and Dr. des. Simon Clematide for their diligent proofreading.

I would like to thank the reviewers, Prof. Dr. Anne Brüggemann-Klein and Prof. Dr. Helmut Schauer, who graciously and promptly agreed to act as reviewers when I contacted them.

I would like to thank my parents and my sister for their constant support and encouragement during all those years.

Finally, and most importantly, I would like to thank my sweetheart, lover, and best friend, Cerstin, for all her support—and for the gentle prodding when I needed it. Her assistance in every respect has been invaluable. As a fellow computational linguist and e-learning expert, her advice has been priceless. I do not know what I would have done without her. *She's there to remind me she believes in me.*

1 Introduction

No magical improvement of material occurs simply because of presentation by a computer-based educational system.

—Avner and Tenczar: *The TUTOR Manual*.

1.1 Why E-Learning?

E-learning is an "unprecedented chance" for universities, according to the *VirtusD Memorandum*, issued in 2007 by a group of twenty-two German politicians, scientists, and entrepreneurs:

> E-Learning gibt uns endlich die bisher nie da gewesene Chance zur Öffnung neuer Wege für die Bildung, zur Steigerung der Bildungsqualität und zugleich zur deutlichen und dauerhaften Rationalisierung des Ressourceneinsatzes in den wesentlichen Bereichen des Bildungswesens.[1] [73, p. 2]

In fact, universities are under pressure, from various sides, to update their teaching and learning technologies from chalkboard and paper to something called *e-learning*.

Universities have to compete with each other for students and for funding, while they face shrinking budgets and reduced staff sizes, and, at the same time, the German government (among others) is trying to increase the number of university graduates in order to meet the

1. "E-learning finally offers us the as yet unprecedented chance to blaze new trails for education, to raise the quality of education, and, at the same time, to significantly and sustainably rationalize the use of resources in the key areas of education." This and all subsequent translations from German are by the author.

demands for highly qualified skilled labor and to keep the country competitive in the global marketplace.[2] As the introductory quotation shows, e-learning is seen as a way to make more efficient use of the available resources.

On the other hand, pressure comes from policy makers, who recognize that "the complexity of the world today requires employees to have extensive knowledge and skills, more than at any other time, to meet the demands of industry and society." [89] In the light of globalization governments see the necessity for a highly qualified workforce to enable their country to compete as a center of research and technology and as a business location on a global level.

In Germany, one visible result is the Excellence Initiative of the federal government and the governments of the *Länder*, which granted a total of 1.9 billion euros for the three lines of funding in two rounds in 2006 and 2007 with the aim of "promoting excellent research in German universities."[3]

Besides increased funding for "clusters of excellence," e-learning is also seen as a crucial factor for success. A call for proposals of the German Federal Ministry of Education and Research (BMBF) from 2004 urges:

> Die systematische Nutzung dieser neuen Technologien ist überdies erforderlich, um das Innovationspotenzial, das sich für Bildungs- und Wissensdienstleister aus dem Trend hin zur Wissensgesellschaft und zur Tertiarisierung ergibt, umfassend auszuschöpfen.[4] [51]

Pressure also comes directly from industry, which demands that graduates be better educated and more highly skilled after shorter studies.

Finally, pressure comes from students: The "Net Generation" [127] demands the same level of service that they know from online businesses, and the availability of e-learning (and integrated e-administration) becomes a selection criterion for potential students and a factor in university rankings. Ulrik Schroeder[5] formulated the following hypothetical questions and demands of "Net Generation" students:[6]

- Why should I have to stand in line to enroll in a course?
- Why should I have to run from one notice board to the next to learn about my grades or get other information?

2. See, for example, [52] or the OECD press release "Deutschland verliert bei der Ausbildung von Hochqualifizierten international weiter an Boden" (September, 9 2008); available online at `http://www.oecd.org/document/15/0,3343,de_34968570_35008930_41277711_1_1_1_1,00.html` (accessed 2008-12-09).
3. `http://www.wissenschaftsrat.de/exini-engl-start.html` (accessed 2008-10-17)
4. "Systematic use of these new technologies is essential to fully realize the potential that arises for education and knowledge providers from the development towards the knowledge society and towards tertiarization."
5. Head of the Computer-Supported Learning Research Group and of the Center for Integrative eTeaching and eLearning Concepts at RWTH Aachen University.
6. From slide 5 of the presentation "eLearning-Einstieg leichtgemacht: L^2P", given in Zurich on 2007-10-31, author's translation.

- Why should I have to go to the examinations office to get some certificate?
- Why should I have to borrow materials and copy them at the copy shop?
- I want to determine time, place, and speed of learning myself!
- I want modern and efficient collaboration with my fellow students!

All this puts pressure on universities to set up infrastructure and to implement institution-wide e-learning.

Since most universities in Europe are government-funded, public funding policies play a significant role as well. The funding policy for university e-learning projects in Germany can be divided into two phases.[7]

During the first phase, from ca. 2000 until 2004, grants were primarily given to single, mostly isolated projects that focused on content creation, especially multimedia-enhanced courseware. The creation of multimedia content is costly—the production of videos and animations, for example, is labor-intensive and requires expensive hardware—and the content is quickly outdated, especially in rapidly evolving subjects. Furthermore, after the end of the project funding, there were typically neither funds nor personnel available for continued maintenance and development of the content produced. Hence, most of these projects had little or no long-term impact.

Consequently, evaluations of this first phase of funding [see 12, 92, 95] highlighted the lack of sustainability as the main problem.

Since about 2005, a new phase of funding policy has begun, which largely follows the recommendations made in the audit report [12]. It was recognized that in order to achieve the sustainable integration of e-learning into universities, it is necessary to consider it as a part of the university development process. The goal is therefore to achieve sustainability by integrating e-learning into the university structures.

Universities have responded to the pressure and to the demands for sustainability by considering e-learning as an issue of high-level strategic planning and devising *e-learning strategies* [see 93]. Thus, as Collis and van der Wende [35] put it, "institutions are now transferring from a period of rich and mostly bottom-up experimentation to a phase in which institution-wide use of ICT [information and communication technology] is being encouraged." We should note that traditional universities are not aiming at replacing face-to-face courses and transforming themselves into distance universities—the use of e-learning is rather envisaged in a *blended learning* approach in which e-learning is conceived as "both complementing and enhancing the overall student learning experience"[8]

7. While we only describe the funding policy in Germany, the development in other European countries has been similar [see 36].
8. From the e-learning strategy of the University of Lancaster `http://www.lancs.ac.uk/celt/celtweb/elearning_policy` (accessed 2008-10-17).

Besides setting strategic goals and defining target numbers, university e-learning strategies typically have two significant implementational aspects: On the institutional side, this is the creation of new institutions such as competence centers or coordination offices. The establishment of an institution-wide technical infrastructure for e-learning constitutes the technical aspect of the strategy. In general, e-learning strategies mandate a *single*, centralized, integrated e-learning platform.

In many cases, this infrastructure is now in place, and practical experience with the systems has been accumulated.

1.2 Problem Statement

Despite heavy investments in software, training, and new institutions, e-learning has up to now failed to affect the transformation of teaching and learning at universities that many have hoped for.

Why is this? We see two types of problems as main causes for this state of affairs, namely *organizational* and *technical* problems. Both types of issues are frequently interrelated.

Many organizational problems stem from the fact that a typical university is not a homogeneous, hierarchical organization but rather a *loosely coupled system of semi-autonomous entities* [182]. Thus, decision-making with respect to the system as a whole can be difficult, and top-down decisions—such as the introduction of e-learning—are likely to face opposition [see 33].

The establishment of new institutions, such as e-learning competence centers—often a part of the e-learning strategy—, may likewise be suspected to be a an attempt at a covert reorganization.

While the decentralized structure of universities may seem antiquated and unproductive, one should keep in mind that universities combine many different subjects, with widely differing contents, course types, and requirements, not to mention different student and faculty personalities and attitudes.

However, when a university-wide e-learning platform is imposed on faculty and students, who were, for the most part, not involved in the selection process, there is the underlying assumption that a single platform can fulfill the needs of all types of users.

These e-learning platforms, often referred to as *learning management systems*, *course management systems*, or *virtual learning environments*, are typically large, complex, and expensive monolithic systems, often requiring training of faculty. This, combined with target figures for the production of e-learning content and unclear specifications of what *e-learning* actually means (see the discussion of the term *e-learning* in section 2.1), is likely to make e-learning appear as a burden to faculty, requiring large amounts of extra work without any perceived benefits.

Grudin [64] pointed out that CSCW (computer-supported collaborative work) applications are likely to fail when they make individuals work merely for the sake of others, without perceived benefit to themselves; university-wide e-learning platforms are comparable systems in many respects.

What is more, the mandated e-learning platform offers a specific, fixed set of tools. This tool set and the overall design of the platform determine the design of courses by making some things easy and other things hard or impossible. On the other hand, it seems that frequently many of the features of the e-learning platform are not used.

To name just a few examples from the literature, Collis and van der Wende [35] note that while most universities and colleges in Europe now have an e-learning platform in place, "the second stage, i.e. rich pedagogical use of this infrastructure, is in many cases still in development." Sclater [157] criticizes that many institutions are only using their e-learning platforms "as little more than repositories for lecture notes and PowerPoint presentations." Likewise, Vovides et al. [179] deplore "many instructors currently use CMSs [course management systems] simply as a delivery mechanism for the subject matter," underutilizing the potential of e-learning to scaffold students' learning. In fact, an Educause[9] study [117] in 2003 found that despite instructors claiming to use the e-learning platform in order to meet pedagogical needs, it seemed that the actual use of the system was mostly for class management needs.

From a technical point of view, the learning management systems (LMS) typically prescribed by universities' e-learning strategies are also problematic. These systems, such as OLAT, Moodle, ILIAS, Blackboard or Desire2Learn, to name a few common ones, are *monolithic systems* with their own databases, their own user management, their own calendars, forums, chats, etc. Since these platforms try to offer a large variety of functions they tend to involve a lot of overhead and a high total cost of ownership. This is compounded by interoperability problems and lock-in effects, i.e., once content is stored in the system it may be hard to move it to another system.

Learning management systems are typically based on certain concepts, which define, for example, what constitutes a *course* and which objects it may contain at which level. These hard-coded structures restrict the choices of instructors and, if they do not match the actual structures, create administrative and usability problems.

Due to predefined, course-oriented navigation structures and other design issues, typical learning management systems cannot be used for managing "general" content, e.g., the university's Web presence. Thus, while both types of content are on the Web, this separates "learning" content from other Web content (which is a distinction that is often hard to make at an institution of learning).

What is more, the content of learning management systems frequently cannot be indexed by Web search engines such as Google. Typically, this is due to issues with access permissions (e.g., the inability of the LMS to grant selective anonymous access) or because content is dynamically created in a way that does not permit crawling. However, in a competitive environment, universities may *want* to make all or selected learning content findable and accessible to the public—high-

9. Educause is a U.S. nonprofit association of more than 2,200 colleges, universities, and educational organizations whose mission is to advance higher education by promoting the intelligent use of information technology. The Educause Web site is at `http://educause.edu/`.

quality educational material can be seen as "promotional" material for both prospective students and funding agencies.[10]

Besides their learning management systems, institutions therefore need to operate, *content management systems* (CMSs) for public content, resulting in a duplication of functionality, increased administrative overhead, and additional maintenance requirements.

1.3 Theses

As we have shown above, there are a number of issues, both organizational and technical, with current conventional learning management systems. This dissertation focuses on the *technical* aspects of e-learning platforms and proposes two main theses.

Current learning management systems offer a large amount of functionality and features in a single, monolithic system. The functionality includes basic system functions such as authentication, user and group management, and resource management (upload and download of content, management of object permissions, search functions, etc.); user-level functionality includes course management (enrollment, student roster, gradebook, statistics), Web page creation and editing, wikis, forums, chats, calendars, drop boxes, quizzes, facilities for questionnaires and surveys, contact forms, e-mail notification, RSS feeds, etc.[11]

There is no question that these features can be useful in e-learning. However, no distinction is apparently made between functionality that is *specific to learning and teaching*, such as assessment, and functionality that can be used in an e-learning context or with learning content, but which is *not* specific to learning and teaching, such as e-mail. In other words, the functionality is not abstracted from its usage domain.

The failure to make this abstraction has numerous negative effects. For example, with many proven and reliable wiki implementations available, why should each e-learning platforms need to reimplement its own wiki from scratch? The effort needed for these reimplementations is effectively a waste of resources, as it duplicates work, while resulting in oftentimes mediocre, unproven implementations. The functionality furthermore tends to be limited and restricted to the usage expected in this particular context.

10. For example, MIT's OpenCourseWare initiative makes MIT course content freely available on the Web. Laudable as it is, this initiative not only wants to "help educators in developing countries, [...] assist students and self-learners," [90] and "make our world a better place" (*President's Message* on `http://ocw.mit.edu` (accessed 2008-10-17)), but it is, of course, also a way to showcase and advertise MIT's "world-class research and world-class teaching." [90, citing former MIT president Charles Vest]

11. For an overview of the functionality of current e-learning platforms see, for example, the *Moodle Features Demo Course* `http://moodle.org/course/view.php?id=34` (accessed 2008-10-17), the *OLAT 5.0 Functional Survey* `http://www.olat.org/website/en/download/OLAT_5_0_Functional_Survey_v3.pdf` (accessed 2008-10-17), or the *Sakai Tool List (2.5)* `http://bugs.sakaiproject.org/confluence/display/DOC/Tool+List+(2.5)` (accessed 2008-10-24).

In fact, more general and more mature implementations of many of the functions not specific to e-learning are already available in content management systems.

Learning objects—educational content, e.g., an explanation of photosynthesis—managed in a learning management system obviously differ in their *purpose* and their *intention* from other content objects typically managed in content management systems, e.g., a news story.

However, if we abstract from the purpose and the intention of the objects (and functionality) found in LMSs and in CMSs, are there any fundamental differences at the *technical* level? We think not. First, a PDF document is a PDF document, a forum is a forum, and a calendar is a calendar—the difference is *only* in the purpose and the intention of the document, the forum, or the calendar. Thus, at the technical level, a forum in which motorcycle tuning is discussed by hobbyists is the same as a forum in which students of a university course discuss interpretations of 17th century poems. Second, the distinction between learning objects and other content objects is blurred by the fact that objects can be repurposed. For example, even though it was originally created for a different purpose, a news item can serve as a learning object in a foreign language or contemporary history course.

Our first thesis is therefore:

Thesis 1: A large portion of the functionality commonly found in e-learning platforms can be abstracted from its domain-specific usage and can be shown to be common content management functionality.

This thesis will be substantiated in chapter 2, where we will discuss definitions of e-learning and review the state of the art in both e-learning platforms and content management.

If thesis (1) is indeed true, it follows that content management systems can be used to provide the basic functionality and, by adding e-learning-specific functionality, extended to serve as e-learning platforms. This will, however, require a modular architecture:

Thesis 2: The architecture of systems that can be used for e-learning has to be modular. Modularity is required on three levels: (1) On the level of implementation, allowing programmers to create new components, (2) on the level of administration, allowing system administrators to specify which components to install and to add third-party modules, and (3) on the level of end users, i.e., allowing instructors to select and combine components to obtain the functionality they need to realize the pedagogical goals via e-learning.

The eduComponents implement these ideas and will serve as a proof of concept for this thesis. Design and implementation of the eduComponents are covered in chapters 3 and 4; chapter 5 shows its successful use through the realization of modularity on all three levels.

1.4 Outline

The remaining chapters are structured as follows:

Chapter 2 ("Foundations and State of the Art") provides the terminological basis and the general technological background for this dissertation. The usage of the term *e-learning* and its various, often disparate, definitions are discussed to eventually arrive at a practical definition for use in this dissertation. It then goes on to provide a short historical overview and a review of the current state of the art of e-learning platforms and content management systems, and it defines what we mean by these terms in the context of this dissertation.

Chapter 3 ("The eduComponents Approach") describes the eduComponents approach in detail, briefly describes the individual components of the eduComponents suite, and discusses related work.

Chapter 4 ("Assessment") focuses on assessment. It outlines the pedagogic background, the motivation for e-assessment and its history. It then describes the functionality provided by the eduComponents and discusses issues of interoperability and authoring of multiple-choice tests.

Chapter 5 ("Practical Use, Experience, and Evaluation") describes the use of the eduComponents in the WDOK research group at Otto von Guericke University Magdeburg and it evaluates the feedback from students gathered through questionnaires over three semesters. This chapter also discusses feedback from users outside the University of Magdeburg.

Chapter 6 ("Outlook") outlines potential directions for new applications of the eduComponents and for further work.

Chapter 7 ("Summary and Conclusion") summarizes the research described in this dissertation.

Notes on Language and Terminology

A number of important terms are defined in the glossary in appendix A. The terms defined there are generally used only with this specific meaning.

The term *user* is particularly ambigous and can have surprisingly many meanings in computer science: In the context of e-learning platforms, it can refer to a student, an instructor, a system administrator, a programmer, or an institution; furthermore, it can also refer to an entity managed by the e-learning platform, identified by a unique ID and associated with certain properties, such as permissions. To avoid confusion, we have tried to use the more specific terms (such as instructor, student, etc.) whenever possible; in the remaining occurrences, we have tried to make the meaning of the term *user* clear through the context in which it occurs.

2 Foundations and State of the Art

2.1 E-Learning

2.1.1 What Is E-Learning?

This dissertation describes research in the area of e-learning and we have frequently used the term *e-learning* in chapter 1. However, *e-learning* is not a well-defined term and means different things to different people.[1] Consequently there is a myriad of definitions; the following definition can be regarded as typical:

> e-Learning is an umbrella term describing any type of learning that depends on or is enhanced by electronic communication online using the latest information and communication technologies (ICT). [122]

At first sight, this definition looks convincing. However, when looking more closely, many such definitions are debatable. For example, by this definition, e-learning requires the use of the *latest* technologies, but it is not defined what this means.

The JISC e-assessment glossary [24] defines e-learning as a *learning process*:

> The process of learning which is supported by the use of ICT (e.g. the Internet, network, standalone computer, interactive whiteboard or portable device). Also used loosely to describe the actual content delivered on-screen, and the more general use of ICT to contribute to learning processes. [24]

1. In fact, there is even disagreement with respect to the spelling: *eLearning*, *e-Learning*, *ELearning*, and *E-learning* can all be observed. We prefer the spelling *e-learning*, in parallel with *e-mail*.

This definition is interesting since it understands e-learning to describe the *learning process*, i.e., what a student is doing while learning; this would mean that e-learning is independent from any institutional use of technology: A single student using a standalone PC for vocabulary drills would thus qualify as e-learning.

However, by this definition, e-learning is not only the learning process but also the delivered content and any educational use of computers.

The American Society for Training & Development (ASTD) has compiled an online e-learning glossary which provides the following definition:

> E-learning (electronic learning): Term covering a wide set of applications and processes, such as Web-based learning, computer-based learning, virtual classrooms, and digital collaboration. It includes the delivery of content via Internet, intranet/extranet (LAN/WAN), audio- and videotape, satellite broadcast, interactive TV, CD-ROM, and more.
> (`http://astd.org/lc/glossary.htm` (accessed 2008-12-09))

This definition regards *e-learning* as superordinate term and concentrates on the *mode of delivery* of content. The list of delivery modes and media appears somewhat haphazard: No obvious selection critera are evident and it suggests a lack of technical understanding (e.g., *LAN* and *WAN* are *not* synonyms for *intranet* and *extranet*). This attempt at a definition leaves the reader wondering if, for example, an educational TV program only qualifies as "e-learning" if it is broadcast via satellite, but not if it is broadcast terrestrially.

The glossary of elearningeuropa.info[2], offers another definition of *e-learning*:

> The use of new multimedia technologies and the Internet to improve the quality of learning by facilitating access to resources and services as well as remote exchanges and collaboration.
> (`http://elearningeuropa.info/?page=glossary` (accessed 2008-10-17))

This definition takes a point of view opposite from that of the JISC definition on the preceding page: Whereas the JISC definition considers e-learning as the "process of learning" with technology support—one might say it is "student-centered"—, this definition views e-learning from a technical or institutional perspective: E-learning here describes not learning supported by technology but the use of technology to support learning. We find it noteworthy that by this definition, e-learning is necessarily an *improvement*.

We have briefly reviewed a number of definitions for *e-learning* to give an impression of the wide variety of definitions, which, effectively, have not much in common except that e-learning somehow involves computers and learning.

2. The elearningeuropa.info portal is an initiative of the European Commission's Directorate-General for Education and Culture. According to the Web site, it aims to "promote the use of ICT for lifelong learning."

This state of terminological disarray is clearly unsatisfactory. But what is more, even the term *e-learning* itself is the object of frequent criticism because some feel that it conveys an incorrect image or is in other ways problematic. For example, Conole et al. [36] quote Melody Thompson[3] as wondering whether the use of the term *e-learning* is not counterproductive to the goals of e-learning practice and research:

> Are we limiting our ability to reach aspired goals by our own terminology, one that effectively obscures many of the elements that desperately need to be addressed by policy? Whereas education is by definition a multi-faceted activity understood to involve a variety of players and activities – teachers and teaching; students and studying; institutions and structures; information, knowledge and, it is hoped, learning – e-learning is a term comprising one letter representing a physical property of technology (e for electronic) and the *hoped-for* outcome (learning) for one participant in the interaction. Given the power of language to constrain our thinking, is our current circumscribed terminology making it increasingly difficult to keep in mind and focus on elements of this expanding activity that, while not readily apparent in the term 'e-learning' itself, must be understood and included when establishing policy and researching the phenomenon? [Emphasis in original] (Melody Thompson, as quoted in [36, p. 52])

While we agree that language may influence thought, we think it is important to point out that there is nothing in the word *education* that implies that it is a "multi-faceted activity," neither are all the involved players and activities "readily apparent"; as Thompson notes, this aspect of *education* is *by definition* (or rather by convention, to be more precise). Thus, even though the term *e-learning* is perhaps, linguistically speaking, more transparent, its meaning is not solely determined by its constituent parts, but it is just as conventional as that of *education*. Even though the term may be unfortunate, we think it is unwarranted to blame problems of policy on words.

Fagerberg and Rekkedal [50] object that learning is an activity or process and shown as a change in a person's perceptions, attitudes or cognitive or physical skills, and, as such, it cannot be "electronic." They go on to denounce the abuse of the term, although we again doubt that this is the term's fault:

> For instance, the term, e-learning, seems to be used to convince users that some supernatural things happens with your brain when you place yourself in front of a computer screen. This miracle is very unlikely as learning (included distance learning) in the real world is mainly hard work. [50]

Fagerberg and Rekkedal then define e-learning as

> interactive learning in which the learning content is available online and provides automatic feedback to the student's learning activities. Online communication with real people may or may not be included, but the focus of e-learning is usually more on the learning content than on communication between learners and tutors. [50]

3. Director of the American Center for the Study of Distance Education at Pennsylvania State University.

Similarly, Dichanz and Ernst [44] try to deduce a definition from the *usage* of the term, especially in commercial contexts, which they seem to find rather frustrating. Eventually they arrive at the following definition:

> Mit E-Learning sind Lehr- oder Informationspakete für die (innerbetriebliche) Weiterbildung gemeint, die den Lernern (Mitarbeitern) auf elektronischem Wege als Online-Produkte oder über CD-Rom angeboten werden und unabhängig von Zeit und Ort verfügbar sind. Sie enthalten überschaubare Einheiten von Sachwissen, Selbsttestelemente und Testbatterie, die schnelles (Selbst)Überprüfen der Lernergebnisse zulassen. Ihr Ziel ist es, kurzfristig benötigtes Wissen »just in time« verfügbar und lernbar anzubieten. Die Ziele liegen durchweg im Bereich einfacher Lernzielebenen.[4] [44]

Based on this observation, Dichanz and Ernst criticize the term *e-learning* as "unclear, confusing, and insincere" and argue that it should be replaced by the "more precise and more sincere" term *ES learning*, for "electronically supported learning":

> Aus diesen Gründen möchten wir nachdrücklich dafür plädieren, den unklaren, verwirrenden und unehrlichen Begriff des E-Learning zu ersetzen durch den präziseren und vor allem ehrlicheren Begriff des ES-Learning, des Electronically supported Learning, der ähnlich wie die Bezeichnungen des CAD (Computer assisted design) und CAC (Computer assisted construction) genauer beschreibt, was elektronisch beim lernen [*sic*] tatsächlich möglich ist.[5] [44]

While *ES learning* may indeed be a more precise term, it would not do away with the problems criticized by the authors (e.g., the restriction to lower cognitive levels).

However, Dichanz and Ernst then go on to *redefine* electronically supported learning:

> Mit Electronically supported learning sind dann Lernprozesse gemeint, die in Lernumgebungen stattfinden, die mithilfe elektronischer Medien gestaltet werden.[6] [44]

Recently there seems to be a tendency to avoid the term *e-learning* altogether. For example, in 2007 the E-Learning Industry Group changed

4. "E-learning describes teaching or information packages for further (corporate) education, which are electronically delivered to learners (employees) as online products or on CD-ROM and which available independently of time and place. They contain easy-to-grasp units of factual knowledge, self testing elements, and test batteries, which allow quick assessment or self assessment. They aim at providing just-in-time access to knowledge required at short notice. The goals are consistently on the lower levels of learning goals."
5. "For these reasons, we would like to emphatically advocate replacing the unclear, confusing, and insincere term e-learing with the more precise and, in particular, more sincere term ES learning, electronically supported learning, which, similarly to the terms CAD (computer-assisted design) and CAC (computer-assisted construction), better describes what is actually possible electronically when learning."
6. "Electronically supported learning then refers to learning processes which take place in learning environments created through electronic media."

its name to *European Learning Industry Group*; in a news article [5], Joe Hegarty, the organization's chairman, is quoted as saying that the term *e-learning* has been "overused" and that "[t]echnology is now clearly embedded in all modern learning solutions."

For example, in the preface to [125] the authors comment the change of title of the book from *Kompendium E-Learning* to *Kompendium multimediales Lernen* ("Compendium Multimedia-Based Learning") in the new edition as follows:

> Auf die Bezeichnung »E-Learning« zu verzichten, fiel nicht allzu schwer. Zum einen handelt es sich ohnehin um ein Label aus dem Marketingbereich und nicht aus der Wissenschaft, zum anderen gelten viele wissenschaftliche Aussagen zu einem multimedial unterstützten Lernen nicht nur für die elektronische Darbietung, für die das »e« steht, sondern ebenso für die Gestaltung von Lehrbüchern und -materialien, Folien usw.[7] [125, p. V]

The book still contains many references to *e-learning*, but the comment is interesting in that Niegemann et al. do *not* see e-learning as a unique mode of learning but rather an extension of more traditional forms of learning. We are not sure, though, in which respect the term *multimedia-based learning* is "more scientific" than *e-learning*, especially since *multimedia* could very well be considered a typical "marketing label."

In 2007, Scott and Vanoirbeek introduced a special issue of *ERCIM News*[8] as follows:

> Initially known as 'computer-assisted learning' and later e-learning, the study of improving learning processes by the use of technology has evolved into the research domain known as 'Technology-Enhanced Learning'. Its objective is to encourage the emergence of new learning models that are sustained by the context-aware use of technology and anchored in the practices of users. [159]

Indeed, *technology-enhanced learning* seems to be the term preferred by the European Commission, it is, for example, the term used in the calls of the Seventh Framework Programme (FP7)[9]. Nevertheless, most articles of the above-mentioned special issue use the term *e-learning*, so it does seem that it is primarily the European Commission that speaks of *technology-enhanced learning*.[10] In their introduction, Scott and Vanoirbeek also offer yet another definition of *e-learning*: Here it is "the study of improving learning processes by the use of technology," i.e., the focus is not on the actual use of the technology, but on the field of research.

As we have already pointed out above, there does not seem to be much consensus about the exact meaning of the term *e-learning*, which is also criticized itself.

7. "It was not hard to dispense with the term 'e-learning.' First, it is a marketing label anyway, not a scientific term, second, many scientific results regarding multimedia-supported learning do not only apply to electronic presentation but to the design of textbooks and teaching material, slides, etc. as well."
8. ERCIM: The European Research Consortium for Informatics and Mathematics.
9. See `http://cordis.europa.eu/fp7/ict/programme/home_en.html` (accessed 2008-10-16)
10. Although the preferred term used to be *eLearning* [see 128].

One problem with defining e-learning is that most definitions are intentionally very broad and inclusive; most authors thus seem to agree that it is a wide field, and the apparent vagueness and variety of definitions of the term therefore only reflect the reality of this field, which includes many different applications and areas of research, with different individuals and groups concentrating on different issues. Nevertheless, all the involved parties do seem to share a feeling of being part of a larger field, and we think that this is reflected in the inclusive definitions of *e-learning*.

While this can be regarded as a good thing, there are actually two main types and views of e-learning, *corporate* and *academic* e-learning, and there are significant differences between the two. Surprisingly, these differences are only rarely mentioned.

2.1.2 Corporate and Academic E-Learning

In corporate settings, e-learning typically refers to the delivery of learning content (courses) to students (employees), with the goal of enabling "just-in-time" learning at the workplace and of quickly reaching all of the involved personnel, which may be geographically dispersed, without costly and time-consuming travel. Time and cost savings are the primary motivation for corporate e-learning.

The courses are thus typically relatively short, highly structured units, which are intended to be used in a distance-learning situation. Typical examples are product training before the introduction of a new product and compliance training in response to new regulatory requirements.

A press release "E-learning: Allianz introduces global platform"[11] from Allianz[12] illustrates this well.

The press release starts by motivating e-learning with an increasing number of people learning "on the job" and decreasing financial resources. It then goes on to quote a project manager, Werner Waldner, elaborating on the advantages of e-learning for the company and underlining the cost savings: "[Students] can learn where and when it suits them and review the material long after they have completed the class at no additional cost."

Companies in the consumer finance and insurance business typically need to have a physical presence where their customers live. This necessitates that part of their workforce is widely dispersed; also, they frequently collaborate with local representatives which are not necessarily employees. Consequently, Waldner highlights that e-learning enables the company to "reach employees, agents, and brokers even in remote areas, and at a cost that makes learning financially viable even for small learning groups."

Waldner is quoted as saying that the focus of the e-learning project is "to improve the time-to-market of new products (product training)

11. http://www.allianz.com/en/allianz_group/press_center/news/company_news/human_resources/news12.html (accessed 2008-10-16)

12. Allianz describes itself as a "service provider in insurance, banking and asset management." According to information on its Web site, the Allianz Group has (as of June 30, 2008) approximately 181,000 employees worldwide and over 80 million customers in about 70 countries.

and meet the demanding regulatory requirements in our industry (compliance training)." As an example for mandatory compliance training the press release cites the U.S. Sarbanes-Oxley Act, which affects corporate governance and financial disclosure for companies listed in the United States. The company needs to ensure that all concerned employees (actuaries, managers, controllers, etc.) have the same knowledge level; if this were to be realized using face-to-face training, it would incur significant training costs.

Another Allianz press release[13] from 2003, entitled "Anytime, anywhere – e-learning at Allianz," makes it even more clear that the primary motivation for corporate e-learning are cost savings. The press release clearly states that the "prospect of low-cost, efficient online training for staff and agents has a lot of appeal to human resource departments." The press release gives numbers from a subsidiary company (Fireman's Fund), where training staff size fell from over 90 to 30 while the number of completed training units doubled. The implementation of e-learning programs is said to have helped to reduce the training budget from 12.8 million U.S. dollars in 2001 to 4.5 million U.S. dollars in 2003, which means that the average cost per completed unit dropped from 1,500 U.S. dollars in 2001 to 389 U.S. dollars in 2003.

Faced with shrinking budgets and staff sizes and rising student numbers, universities[14] also aim to reduce the costs of education. E-learning is hoped to make teaching and learning more efficient, more effective, and more flexible. Nevertheless, there are crucial differences between corporate and academic e-learning.

The most important one is probably: Education is a university's "core business," whereas for a company it is only an auxiliary activity—in business terms, for a university it is a *profit center*, while it is a *cost center* for a company. A university (or another learning institution) introducing e-learning is changing its way of doing "business." This is further complicated by the fact that, as Françoise Caillods notes in her preface to [11], "Education cannot be looked upon as a mere product to be bought and sold in the same way as one would buy a book, a compact disc or a car." [11, p. 9]

Universities also differ from companies in their organizational structures. As we have noted above (see section 1.2), most universities are not homogeneous and hierarchical but are rather loosely coupled systems of semi-autonomous entities. Thus, decision-making with respect to the system as a whole can be difficult and there is no authority that could easily force the subsequent university-wide implementation of decisions. Kubicek et al. [95] specifically discuss problems and possible solutions concerning the introduction of e-learning at German universities.

Another difference is the role of students: At a university, the students are *customers;* they pay (directly or indirectly) for a service, they have voluntarily chosen to learn. At a company, the students are employees; they are paid to do their job, which may include learning to get better

13. http://www.allianz.com/en/allianz_group/press_center/news/company_news/human_resources/news18.html (accessed 2008-10-16)
14. We use the term *universities* to also cover similar institutions of higher education.

at their job. In many cases, learning is not voluntary but mandatory (see the example of compliance training above).

This difference has further consequences. As we have already mentioned in section 1.1, students—regardless of whether one wants to assume a "Net Generation" or not—will increasingly expect the same level of service, convenience, and "connectedness" they are used from other activities, such as shopping, communicating, or accessing information. The availability of e-learning (and integrated e-administration) thus becomes a selection criterion for potential students—and instructors—and a factor in university rankings.

This puts pressure on universities to implement e-learning and associated infrastructure; consequently, an attractive e-learning offering will then constitute a direct competitive advantage for a university in attracting students and faculty. For companies, on the other hand, e-learning can only be an indirect advantage as customers may profit from cost savings or better service.

As university members are often involved both in education and in research, they are much more likely to explore new ways of teaching and learning, whereas companies are generally only interested in changes if they offer better training at a lower cost.

This brings us to another important difference: Universities primarily offer *education*—"intellectual, moral, and social instruction" (*New Oxford American Dictionary*, 2nd ed.). Companies, on the other hand, are mostly interested in *training*, i.e., teaching a "particular skill or type of behavior," which can then be immediately applied in their job, e.g., how to operate a particular machine or how to determine whether a cross-border financial trade they have been offered is legal and compliant with corporate policies and ethics.

A further point is that companies are frequently buying e-learning content from publishers for training in areas that are not company-specific. Universities, on the other hand, are usually producing their own learning materials, or rather, individual instructors are creating the materials for their own courses. In Europe, reliance on text books is much lower than in the United States, and e-learning content is even more rarely acquired from external sources.

A white paper [135] by Perdisco, an e-learning content publisher, therefore characterizes corporate e-learning as *content-focused* and academic e-learning as *technology-focused*, meaning that universities are focused on "enabling technologies" such as e-learning platforms, but rely on instructors to create content. Being written by a publisher, the white paper is, of course, exhorting universities to buy more e-learning content; it is debatable whether this is desirable. Nevertheless, the characterization is certainly fitting.

However, the situation in academic e-learning is more complex than that. Besides the structural differences listed above, most universities are publicly funded and are thus subject of political discussion and of political decisions. This does not only concern budgetary matters, but influences university policies in other ways as well: "Education transmits values, contributes to forging national identity, and to

strengthening national integration and solidarity" [11, p. 9] and is thus a matter of public interest.

For example, as we have said before, universities do hope for cost savings through e-learning, especially to handle larger numbers of students with less staff. However, developing e-learning in universities has in fact proven to require significant upfront investments—Bates warns: "E-learning is not a cheap alternative to face-to-face teaching" [11, p. 83]. Furthermore, since education is a matter of public interest, governments and politicians usually avoid talking about cost-effectiveness in the context of education. Thus, while increased efficiency is typically mentioned as a secondary goal, the main emphasis is generally put on the improvement of the *quality* of instruction.

The following excerpt from a report by the Office of Technology Assessment at the German Parliament (TAB) lists some of the goals and expectations associated with academic e-learning, without even mentioning cost issues:

> Der Nutzen für die Lernenden bzw. der Mehrwert wird in der flexiblen, zeit- und ortsunabhängigen Nutzung gesehen, in der größeren Motivation, durch neue Lernszenarien und kommunikative, interaktive Betreuung zu lernen, in Möglichkeiten zur Simulation realer Situationen, in vielfältigen, auch kollaborativen Gestaltungsoptionen sowie in der Möglichkeit, ergänzend Informationen oder Wissensbausteine zu nutzen bzw. zur Verfügung stellen zu können.[15] [144, p. 5]

The following quotation from a call for proposals of the German Federal Ministry of Education and Research (BMBF) also illustrates the expectations for quality improvements:

> Erwartet werden nicht nur eine deutlich verbesserte Qualität von Lehren und Lernen, sondern auch eine deutlich erhöhte Effizienz der Wissensvermittlung und eine bessere Vorbereitung auf die Erfordernisse des lebenslangen Lernens.[16] [51]

Finally, to give an example from an other country, Lepori and Succi [98] list the following objectives for the introduction of e-learning into Swiss universities:

1. to improve the education quality
2. to increase accessibility and flexibility of curricula
3. to answer the technology trend arising in the Higher Education sector

15. "We see the advantages for learners, or the added value, in the flexible, time- and location-independent utilization, in the higher motivation to learn through new learning scenarios and through communicative, interactive tutoring, in the feasibility of simulating real situations, in the numerous design options, including collaboration, and in the potential for using and offering supplementary information or knowledge blocks."

16. "We not only expect a significant quality improvement in teaching and learning but also significantly raised efficiency of knowledge transfer and better preparation for the requirements of life-long learning."

Table 2.1: Comparison of selected features and their values in corporate and academic e-learning.

	Corporate	Academic
	Institutional characteristics	
Institution type	Companies	Universities
Core business	Anything but education	Education
Primary motivation	Cost savings	Quality improvements
	Learner characteristics	
Target group	Employees	Students
Learner role	Employee, subject to directives	Customer, voluntary
	Course characteristics	
Goal	Training	Education
Typical scenario	Distance learning	Blended learning
Learner situation	Individual	Class-oriented
Unit size	Short (hour-long)	Long (up to a semester)
Topics per unit	One	Multiple
Course structure	Sequential	Varying
Communication among learners	Less important	Important

4. to improve the efficiency of investments
5. to reduce costs
6. to enhance skills needed in lifelong learning

Again, quality improvement is named as a goal before cost reductions.

To summarize: While academic e-learning shares some aspects with corporate e-learning, it is in effect quite different. Table 2.1 is a summary of some of the differences between corporate and academic e-learning.

The motivation for corporate e-learning is straightforward: Many employees can receive training at once, and it enables training to be flexibly incorporated into the work day of the employees, wherever they are, whereas traditional classroom-based training is disruptive and expensive; it may also not be possible if many employees need to be trained in a very short period of time, e.g., in the case of compliance training, as the personnel, the facilities, and common allocatable time may not be available. Once e-learning courses are available, they can also be reused, for example, for training new employees. Thus, there is a clear business case for e-learning.

The motivation for academic e-learning is much more faceted. This is due to significant differences between universities and companies, as well as the fact that many more stakeholders with different interests are involved. As education is the "product" of universities, e-learning is, in general, not only expected to improve the cost-effectiveness, but also the *quality* of education.

These two types of e-learning obviously cannot—and must not—be treated as identical or interchangeable, as they are in many respects

hardly comparable. Unfortunately, people not familiar with both types of e-learning are prone to confusing them. For example, Dichanz and Ernst's problems with the term *e-learning* (see the discussion on page 28) are a good case in point.

2.1.3 E-Learning: A Practical Definition

As we have seen, it is not easy to come up with a good definition for the term *e-learning*—or, for that matter, with a better term. We will therefore stick to the term *e-learning*, as it seems to be neutral and most widely used—or, as Jones [84] put it:

> 'e-learning' is one of many terms currently used to describe the use of information technology to support teaching and learning. Rather than argue about the ambiguities and differences among the various terms, this paper will use 'e-learning'. [84, p. 54]

We will use *e-learning* according to the following definition:

Definition: E-learning *is a general term describing all kinds of computer-mediated and computer-supported learning and teaching, regardless of whether it is used exclusively, as in distance learning situations, or complementarily with face-to-face, instructor-led courses (blended learning). In this dissertation, our focus is specifically on academic environments and network-based systems.*

As Lepori and Succi note,

> every term originates from a given moment of the technological evolution, and quite often stems from specific theoretical, pedagogical and even philosophical assumptions. [98, p. 19]

Nevertheless, we consider terms such as *computer-assisted learning*, *technology-based distributed learning*, or *technology-enhanced learning* as quasi-synonymous, as we are not aware of any consistently made semantic distinctions.

However, we would specifically exclude *computer-based training* and *Web-based training*, as these terms are typically applied to systems and content for corporate training, not academic education.

Terms such as *computer-supported cooperative learning*, on the other hand, may be considered to refer to specific forms of e-learning or specific systems, in this case focusing on or promoting collaboration between students.

2.2 Software for E-Learning

We have seen in section 2.1 that e-learning covers many different scenarios and a wide variety of activities. Since e-learning is, by definition, computer-supported, it requires hardware, software, and network infrastructure.

Most e-learning environments today are *Web-based*, i.e., they are accessed via Web browsers (using HTTP) over a TCP/IP network such as the Internet or an intranet (e.g., a university campus network).

Thus, in general, e-learning today does not have any special hardware or networking requirements: In theory, only Internet access and a computer capable of running a Web browser is necessary to access Web-based e-learning applications. In practice, many applications make use of client-side scripting (using JavaScript, Adobe Flash or Java Applets) or contain media or documents requiring proprietary software (such as Apple QuickTime or Microsoft Windows Media players for movies or Microsoft PowerPoint for presentations), so that a certain amount of computing power must be available and the choice of operating systems may be restricted.

An institution offering e-learning also only needs standard server hardware and Internet connectivity, both of which must, of course, be sized according to demand (i.e., it depends on factors such as the number of students simultaneously using the system and the type and amount of media being served).

Many types of software and network services can be used for e-learning; examples include e-mail, Usenet, chats, discussion forums, blogs, collaboration (CSCW) tools, simulation software, testing and assessment software, e-portfolios, vocabulary trainers, and games.

These applications can be used individually or in various combination for e-learning; for example, Graziadei [63] describes a "Virtual Instructional Classroom Environment in Science" from the early 1990s based on a variety of programs and services, including e-mail, VAX Notes[17], Questionmark Perception, and HyperCard[18].

However, the drawbacks of setups like this are obvious: The most serious are the lack of common user management and authentication, varying user interfaces, and limited interoperability between the tools. With the advent of the Web and the institutionalization of e-learning, Web-based *e-learning platforms* were created to provide a single, consistent user interface for all aspects of a course.

The functionality of e-learning platforms typically includes access to learning content and tests and communication and collaboration tools for students, and course management and assessment facilities for instructors. E-learning platforms may also include administrative functionality or interfaces to administrative systems (often called *campus management systems*) for managing student admissions and enrollment (sometimes termed "student lifecycle management"), for resource planning, accounting, etc.

17. VAX Notes, later DECnotes was an early computer conferencing system—inspired by the Notes system of PLATO (see below)—developed in the 1980s at Digital Equipment Corp. [see 4]

18. HyperCard, first released in 1987, was a hypermedia programming environment by Apple Computer, Inc. It was one of the first widely used hypertext systems before the Web.

2.2.1 What Is an E-Learning Platform?

Up to now, we have used the term *e-learning platform* without properly defining it. Unfortunately, there is as much terminological disagreement about how to name these software systems as there is about the term *e-learning* itself.

Consequently, there are many names for software systems facilitating or supporting e-learning, such as *learning management system* (LMS), *learning content management system* (LCMS), *course management system* (CMS), *virtual learning environment* (VLE), *managed learning environment* (MLE), or *learning support system* (LSS). Usage is not consistent: Depending on the author (or the vendor), some of these terms may describe different types of systems, or they may be used more or less interchangeably.

We prefer the term *e-learning platform*; first, it shows the relationship to the activity it is supposed to facilitate or support, i.e., e-learning. Second, the term *platform* is generic enough not to suggest a certain implementation or structure, and, unlike *management system*, it does not put the focus on managing.

A specific problem with the term *course management system* is that it shares the abbreviation "CMS" with *content management system*. This is confusing for people familiar with the latter, much more frequently used meaning, and it would be especially problematic in this dissertation, where we are talking about *both* e-learning platforms and content management systems.

2.2.2 Some Historical Notes

It may be assumed that e-learning platforms are a new phenomenon which appeared with the spread of the Web. In fact, today there is an implicit understanding that an e-learning platform has a Web user interface; this is consistent with the general assumption that e-learning *per se* is Web-based.

However, the ancestry of today's e-learning platforms can be traced back to earlier computer-assisted instruction (CAI) systems which started to be developed in the 1960s. Early such systems, e.g., PLATO I [18], are certainly not e-learning platforms in the modern sense, but later mainframe-based systems pioneered a number of key concepts. Especially noteworthy are the later versions of PLATO, PLATO III and PLATO IV.[19]

The PLATO system was originally developed at the University of Illinois; from 1976 until the late 1980s it was marketed commercially by Control Data Corporation. The PLATO project received funding from the U.S. National Science Foundation (NSF) and was formally evaluated by the Educational Testing Service (ETS) in 1978 [120].

PLATO ran on Control Data Cyber 70 series mainframes, serving up to 1000 concurrent users, and used highly innovative, microprocessor-based terminals with bitmapped plasma graphics displays, touch screens, and local processing capabilities (see figure 2.1 on the next page). At the time, most terminals were *dumb*, i.e., they were hard-

19. PLATO stands for "Programmed Logic for Automatic Teaching Operations."

Figure 2.1: The microprocessor-based PLATO V terminals for the PLATO IV system featured plasma displays, infrared touch panels, and local processing capabilities. (Photography contributed to Wikipedia by user Mtnman79 and licensed under the Creative Commons Attribution 3.0 License)

wired to interpret only a limited number of simple control codes, without any programmability and local computing capabilities. The PLATO V terminal, on the other hand, was based on an Intel 8080 microprocessor [169] and was thus able to execute programs downloaded from the host computer—this is very similar to how today's Web browsers execute ECMAScript (JavaScript) code downloaded from a Web server.

The plasma display was, in fact, also an invention from the Computer-Based Education Research Laboratory (CERL) at the University of Illinois, the group developing the PLATO system [19]. For audio output, a synthesizer could be connected to the terminal. The PLATO synthesizer developed by Sherwin Gooch was one of the first computer-controlled music synthesizers [60] and an innovation in itself. The synthesizer was used for a variety of educational applications, such as melodic dictation in music education [134] and text to speech (TTS) for language teaching [163].

Regarding software, PLATO introduced concepts such as online forums and message boards, online testing, email, chat rooms, instant messaging, remote screen sharing, and multi-player online games.

While the origins of the PLATO project were in programmed instruction, PLATO III and PLATO IV were not dedicated to a single paradigm of instruction. The availability of the TUTOR programming language [9] certainly played an important role, as it allowed instructors to design their own lessons with relative ease. At the same time, it was effectively a general-purpose language, so it could be used to implement any approach considered adequate, or for writing software that was not directly learning-related, including (non-educational) games.

While it is rarely acknowledged, PLATO can, in many respects, be considered a precursor of today's e-learning platforms.

2.2.3 E-Learning Platforms: A Practical Definition

We will now try to find a practical definition for the term *e-learning platform*. Concerning the functionality of e-learning platforms, Collis and Moonen [34], referring to Robson [148], define a *WWW-based course-management systems* as

> comprehensive software package that supports some or all aspects of course preparation, delivery and interaction, and allows these aspects to be accessible via a network. [34, p. 78]

The vagueness of this definition indicates that it is hard to give a precise definition of what an e-learning platform exactly is. Nevertheless, most researchers and practitioners are likely to agree on a set of *typical* e-learning platforms; this set would probably include Blackboard and Moodle, the currently most popular platforms. Thus, while there clearly is a class of software systems identifiable as "e-learning platforms," it is not defined by a "checklist" of necessary and sufficient features which a system must possess in order to be an instance of the class. We thus rather have to take a "prototype view" (in the sense of Fillmore [54]), i.e., e-learning platforms—as a concept—are characterized by a number of properties—not all of which are equally important—which actual systems fulfill to various degrees, so that membership in the class of e-learning platforms is gradual.

Robson [148] lists five common features of "course-support systems": computer-mediated communication, navigational tools, course management, assessment, and authoring tools, and gives examples for these features, which we will briefly summarize:

Computer-mediated communication to allow students and instructors to communicate with each other. Examples are discussion forums, messaging systems and chats.

Navigational tools organizing a site into units such as modules and lessons and granting access to the various features of the system.

Course management for keeping track of students and their records and for managing security and access rights for various user groups.

Assessment is usually provided in the form of online quizzes with immediate feedback, including a score and comments.

Authoring tools: Most course-support systems do not offer full authoring environments, but allow instructors to upload and organize material, to create discussions and quizzes, and to generally control the features offered by the system.

De Boer [41] describes an e-learning platform[20] as an

> integrated combination of Web-based tools specifically focused on the educational support of distributing content and enabling communication and organization and pedagogical support within courses. [41, p. 23]

To describe the differences between different systems, de Boer considers the design and development of e-learning platforms to be affected by four "lines of influence": (1) Communication systems (e.g., e-mail and chats), (2) knowledge management systems, (3) computer-based training, and (4) computer-supported collaborative work (CSCW).

From these four sets of influences, he derives four "dimensions"[21] and visualizes them as quadrants of a system of coordinates, where the "dimensions" represent types of activities: Communicating, organizing, delivering, and creating. Different e-learning platforms can then be thought of as being located at different points in this system and covering different areas. However, his diagrams are only rough sketches and only serve as an explanation of the general idea. We also think that by concentrating on influences from non-e-learning applications, de Boer misses some of the activities specific to e-learning. We consider two important activities to be missing in de Boer's description, namely *collaboration* (which is also missing from Robson's list) and *assessment* (which Robson does list).

While de Boer names CSCW systems as an influence on e-learning platforms, he does not consider collaboration as an activity on its own. We, however, think that collaboration is a very specific activity not covered by de Boer's four activities.

De Boer's dimensions for the most part match the features given by Robson [148] (see above), and in fact he maps Robson's features to his four "dimensions" [see 41, p. 27]. Here Robson's assessment feature is assigned to the dimensions "creation" and "content delivery." While assessment does involve creation and delivery of tests, it is much more than that and it is an important educational activity.

For research into the use of e-learning platforms, Malikowski et al. [107] propose a model for describing and evaluating features. Their model consists of five categories:

1. Transmitting course content, i.e., delivery of content and course-related information, including grades,

20. De Boer uses the term *course-management system*.
21. Unfortunately, de Boer's terminology is somewhat inconsistent: He variously refers to the "dimensions" as "structure," "characteristics," or "lines of influence."

2. evaluating students, i.e., assessment,

3. evaluating courses and instructors, i.e., surveys and questionnaires,

4. creating class discussions, i.e., communication functionality such as forums, and

5. creating computer-based instruction, i.e., facilities for sequencing learning content.

Malikowski et al. do not attempt to give an abstract definition of what an e-learning platform[22] is, but rather describe the features found in current systems (Blackboard, Desire2Learn, and WebCT are explicitly mentioned). The five categories of their model can be partially matched to Robson's features and de Boer's activities; Malikowski et al. specifically list the evaluation of courses and instructors and category 5, "creating computer-based instruction," combines a number of functions from different items on Robson's list (primarily from navigational tools, assessment, and authoring tools). On the other hand, Malikowski et al. do not have a category covering student collaboration—only a small portion of collaboration may be described by category 4, "creating class discussions."

Based on the descriptions and definitions of e-learning platforms discussed above, we suggest *six activities* that characterize e-learning platforms: *Creation*, *organization*, *delivery*, *communication*, *collaboration*, and *assessment*.

Similar to Robson [148], we may briefly define these activities as follows:

Creation refers to the production of learning and teaching materials by instructors.

Organization refers to the arrangement of the materials for educational purposes (e.g., combining them into modules or courses).

Delivery refers to the publication and presentation of the materials, so that they can be accessed by students.

Communication refers to the computer-mediated communication between students and instructors and among students.

Collaboration refers to students jointly working on files or projects; it also includes collaboration between instructors.

Assessment refers to the formative and summative evaluation of learning progress and outcomes, including feedback.

As we have said before, e-learning platforms support these activities to varying degrees. However, to be classifiable as an e-learning platform, a system will need to support most of these activities to some degree, while other systems are likely to support only few of these activities.

22. Malikowski et al. use the term *course management system*.

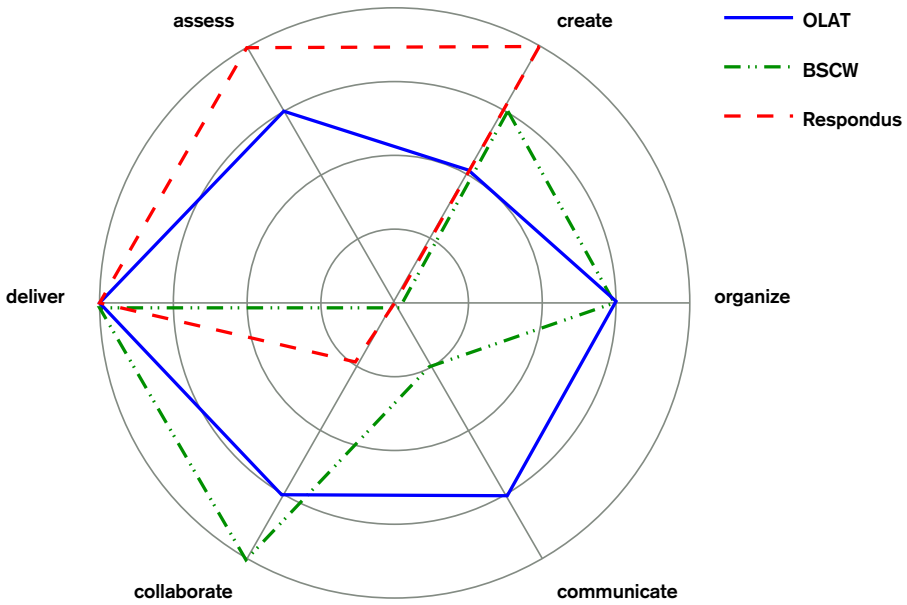


Figure 2.2: Radar plots can be used to visualize the level of support systems provide for different activities. Here, approximate graphs for an e-learning platform (OLAT, see section 2.5.1), a CSCW system not oriented towards e-learning (BSCW [6]), and an assessment tool (Respondus) are given. The radial coordinates should be read as "no support" at the center and "extensive support" at the outermost circle.

This can be visualized using a radar (or spider) plot, as shown in figure 2.2: The six axes represent the various activities that e-learning platforms typically support. When graphing actual systems, each system is characterized by a shape which depends on the level of support for each activity.

As can be seen from the graphs in figure 2.2, the e-learning platform offers medium to high support for all of the activities, while other systems only provide support for a few of the activities.

Building on our preceding analysis, we thus arrive at the following definition:

Definition: *An e-learning platform is a system which provides* integrated *support for the six activities—creation, organization, delivery, communication, collaboration, and assessment—in an educational context.*

Thus, we understand the term *e-learning platform* to describe systems which offer more than one service, covering a fair percentage of the facilities required for e-learning.

The term *integrated* here means that, regardless of the *actual* implementation, an e-learning platform must function as *one* system and it must appear as a single system to the users, i.e., if it consists of separate subsystems, end users should not be aware of the individual subsystems; in any case, data exchange between the constituent elements of the e-learning platform must be fully automatic.

A definition similar to our own is given by the U.K. Department for Children, Schools and Families[23]; the term *learning platform* is defined as follows:

> It is an umbrella term that describes a broad range of ICT systems used to deliver and support learning. As a minimum, we expect it to combine communication and collaboration tools, secure

23. Formerly the Department for Education and Skills.

> individual online working space, tools to enable teachers to manage and tailor content to user needs, pupil progress tracking and anytime/anywhere access. You might hear the term learning platform being applied to a virtual learning environment (VLE) or to the components of a managed learning environment (MLE).
>
> [176, p. 18]

For example, an application offering *only* multiple-choice tests (such as Hot Potatoes) is, according to our definition, not an e-learning platform. A loose collection of tools, such as the ELBA E-Learning Toolbox[24], which is offered to teaching staff at ETH Zurich and the University of Zurich, also does not qualify as an e-learning platform. Instructors using tools from the ELBA E-Learning Toolbox have a single administrative point of contact (the team of the "Network for Educational Technology" (NET); [see 160]), the constituent tools (including, among others, the above-mentioned Hot Potatoes tool for quizzes, Phorum as a discussion board, and Survey, developed at the Virginia Polytechnic Institute, for Web-based surveys) are in no way technically integrated: Every tool has its own user management and no data is shared.

2.3 Do We Really Need Special Software for E-Learning?

After evaluating a number of different definitions, we have presented our own definition of e-learning platforms—which is based on six characteristic activities—in the preceding section.

Let us now consider the following list of functions, which, according to Schulmeister [156], define e-learning platforms:[25]

> - Eine Benutzerverwaltung (Anmeldung mit Verschlüsselung)
> - Eine Kursverwaltung (Kurse, Verwaltung der Inhalte, Dateiverwaltung)
> - Eine Rollen- und Rechtevergabe mit differenzierten Rechten
> - Kommunikationsmethoden (Chat, Foren) und Werkzeuge für das Lernen (Whiteboard, Notizbuch, Annotationen, Kalender etc.)
> - Die Darstellung der Kursinhalte, Lernobjekte und Medien in einem netzwerkfähigen Browser.
>
> [156, p. 10]

We can map these functions to the activities of our definition (see p. 42): The first three items are related to organization; the fourth item is communication, and last item is delivery.

24. `http://www.elba.ethz.ch/index_EN` (accessed 2008-10-17)
25. Translation:
    - User management (authentication with encryption)
    - Course management (courses, content management, file management)
    - Management of roles and permissions with fine-grained permissions
    - Communications facilities (chats, forums) and learning tools (whiteboard, notepad, annotations, calendar, etc.)
    - The presentation of course contents, learning objects, and media in a network-capable browser.

Schulmeister's list thus only covers three of our six activities. What is more, apart from course management proper, the list contains only typical basic content management functions (e.g., user, permissions, and, obviously, content management) and a number of functions commonly available in Web content management systems (either built-in or as add-ons), such as forums or calendars.

Most interestingly, Schulmeister does not mention assessment functionality in this list—since some of the systems he considers in his review do not even offer assessment facilities—, but he lists assessment tools (along with tools for authoring, conferencing, collaboration, reporting and statistics, which are, again, not specific to e-learning) as "important additions" to e-learning platforms [156, p. 101].

The similarities between content management systems and e-learning platforms are also noted by Bergstedt et al. [16]. What is more, the authors identify significant shortcomings in e-learning platforms compared to CMSs, namely that content is usually stored in the form of self-contained units combining content *and* presentation. This approach is embedded in many standards for e-learning data interchange, such as SCORM [1], the Sharable Content Object Reference Model. It is thus not possible to ensure a consistent "look" for all course material on an e-learning platform or to personalize content.[26]

The University of Zurich has been using OLAT as its "strategic learning management system" since 2004. E-learning is used to support face-to-face learning and for preparation and follow-up work, but there are no e-learning-only (distance) courses. Such usage of e-learning platforms can thus be considered typical for traditional universities.

According to Hans-Jörg Zuberbühler, Head of the Customers & E-Content Team of the Multimedia and E-Learning Services, which operate OLAT at the University of Zurich, OLAT usage concentrates on the following functions [Personal communication, 2008-09-24]:

1. Administration of participants and groups, e.g., to handle enrollment for labs, excursions, tutoring courses, etc.
2. Communication and collaboration functions, such as forums, chats, and wikis.
3. Publication of learning content, either as individual HTML pages or as IMS CP content packages [70]. While SCORM is supported by OLAT, it is rarely used and not recommended to instructors.
4. Distribution of other course-related documents, using the upload, download, and commenting functions.
5. Assessment using multiple-choice tests.

26. A SCORM Sharable Content Object (SCO) can technically adapt its appearance and behavior for different users. This is, however, a property of the individual SCO, not the e-learning platform, and is neither available in all SCOs, nor can it be controlled from the outside, as SCOs combine content, presentation, and behavior into a single "black box." SCORM is heavily based on the corporate view of e-learning (see section 2.1.2) and considers e-learning platforms (*learning management systems* in SCORM terminology) as operating environments similar to computer operating systems providing certain basic services (such as data storage and authentication) through a standardized API (the SCORM Run-Time Environment (RTE)) and enabling learners to launch SCOs.

These functions clearly parallel those listed by Schulmeister, with the addition of assessment. Administration of participants and groups (item 1 on the facing page) can be considered a function of user and course management.

Malikowski et al. [107] summarizes research into which categories of e-learning platform functionality are used most by instructors in resident (i.e., non-distance learning) courses. They report that the types of functions used most are (1) the transmission of documents to students, (2) asynchronous communications (forums), (3) quizzes, (4) drop boxes (enabling students to submit documents), and (5) surveys. Asynchronous discussions and quizzes each are only used by about 25% of the instructors, thus the delivery of documents is clearly the single most widely used function of e-learning platforms.

Malikowski [106] describe a study on the use of *multiple* functions by instructors; it was found that most of the instructors (60%) used only one or two of the features offered by the e-learning platform (Desire2Learn). Furthermore, Malikowski found that using the e-learning platform only to transmit files (publication of documents, upload and download) occurred over three times as often than using most other features. Also relatively frequent was the transmission of grade information to students, which is effectively also publication. Interactive features, such as quizzes and discussion forums, were only used to a small extent. Thus, it was found that the e-learning platform was primarily used for the publication of information and for the transmission of files.

We can therefore conclude that e-learning can, in fact, be considered as content management *plus* communication/collaboration and assessment functionality. Correspondingly, these are the functions offered by typical e-learning platforms. However, the content management functionality of e-learning platforms is usually only rudimentary (e.g., there is typically no support for versioning or workflow management).

Web content management systems, on the other hand, have long provided more advanced content management features, and many of them also include communication and collaboration facilities. In the following section we will therefore have a look at CMSs.

2.4 Content Management Systems

In this section we will briefly discuss what is generally understood under the term *content management system* to arrive at a definition suitable in the context of this dissertation.

We should first note that when we are using the term *CMS*, we use it with its most frequent meaning as referring to *Web content management systems*.[27] While the term *content management* also encompasses the management of non-Web content (see figure 2.3 on the next page), Web content management is arguably the most widespread type of content management today, and it is the only type relevant in the context of this dissertation.

27. The abbreviation *WCMS* can sometimes be seen.

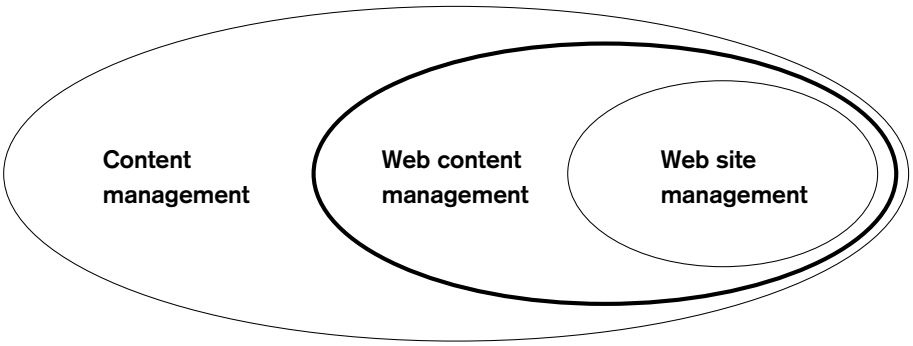


Figure 2.3: Demarcation of Web content management (adapted from [79, p. 160]).

CMSs are used today by practically every company, institution, or individual maintaining a presence on the Web. As soon as there a more than a handful of static Web pages and a single maintainer, the use of a CMS is effectively required.

A variety of definitions for *content management* and *CMS* can be found in books, articles, and on the Web. For the purposes of this paper, an attempt will be made to arrive at a single definition.

Jablonski et al. [79] describe a Web content management system as "a tool for collecting, creating, editing, administering, and publishing content on the Web." [79, p. 160]

Bergstedt et al. [16] note that content management systems evolved from existing systems and thus focus on different functional aspects depending on their heritage (from document management systems, editorial process management systems, workflow management systems, or database management systems). While this gives the (incorrect) impression that all CMSs are based on older systems, it is certainly true that CMSs vary widely in the type and complexity of the functions they implement.

A frequently referenced online definition is found on the Web site *Enterprise Content Management*[28]:

> A CMS is a tool that enables a variety of (centralised) technical and (de-centralised) non technical staff to create, edit, manage and finally publish (in a number of formats) a variety of content (such as text, graphics, video, documents etc), whilst being constrained by a centralised set of rules, process and workflows that ensure coherent, validated electronic content.
>
> (`http://contentmanager.eu.com/cms.htm` (accessed 2008-10-17))

An aspect central to almost all descriptions of CMSs is the *separation of content, structure, and presentation* (see, e.g., [56, p. 161] or [124]), i.e., the presentation of content can be controlled independently from the content itself—this means that the appearance of a Web site can be completely changed without having to touch the content and ensures a consistent appearance across the site. This separation is achieved through *templates* and *style sheets*: Templates specify the general structure of a page (i.e., the document type) and are instantiated by linking the "slots" of the template to specific assets (a text, an image, etc.) or via dynamic queries to the content database, e.g., for the title of

28. `http://contentmanager.eu.com/` (accessed 2008-10-17)

the most frequently requested article. The concrete visual appearance of the various classes of elements is defined in stylesheets, specifying, for example, "Helvetica 16/18, black on white, flush left" for elements of the "title" class.

Friedlein notes that "most CMSs have gone beyond pure content management." He points out that, besides basic and advanced content management features, a "CMS can use its database-driven infrastructure to offer a suite of tools and applications that aim to provide enhanced customer relationship management (CRM) opportunities," in particular personalization and customization and user-adaptive content and navigation. He concludes that "the CMS manages users, profiles, and site data, as well as content assets." [56, pp. 161–162]

While all of these definitions differ in their wording and their focus, there is still a certain amount of overlap. Thus, like e-learning platforms, the class of *content management systems* is only vaguely defined by a certain prototypical set of core functionality.

Based on the references cited above and on other sources [79, 16, 145, 124, 56] we arrive at the following definition:

Definition: *A content management system is a system for supporting the creation, management, and publication of content. The core set of functionality of CMSs comprises (1) user and access management, (2) asset management (including versioning), (3) workflow management, (4) search and retrieval, and (5) a template-based publication facility, providing separation of content, structure, and presentation.*

Besides this core functionality, other typical functions include WYSIWYG editors, facilities for managing parallel content in multiple languages, time-scheduled publication, import/export interfaces, link checkers, statistics, syndication, and many others. Many systems also provide interfaces for extensions or plug-ins.[29]

Figure 2.4 on the next page shows a screenshot of a typical Web CMS user interface. Authors and editors can navigate the site structure using a "page tree," insert and delete content, and edit content elements and their metadata and permissions, through forms and editors.

We have now defined e-learning platforms and content management systems. The following section will describe the current state of the art of these two types of systems.

2.5 State of the Art

We have identified two types of systems which are relevant in the context of this dissertation: *E-learning platforms*, which present themselves as "tailor-made" solutions for e-learning, and *content management systems*, which effectively cover much of the same functionality as e-learning platforms. In this section, we will have a look at the current state of the art of these two types of software.

29. Jablonski et al. [79], p. 166 and Bergstedt et al. [16], among others, even consider this an essential feature.

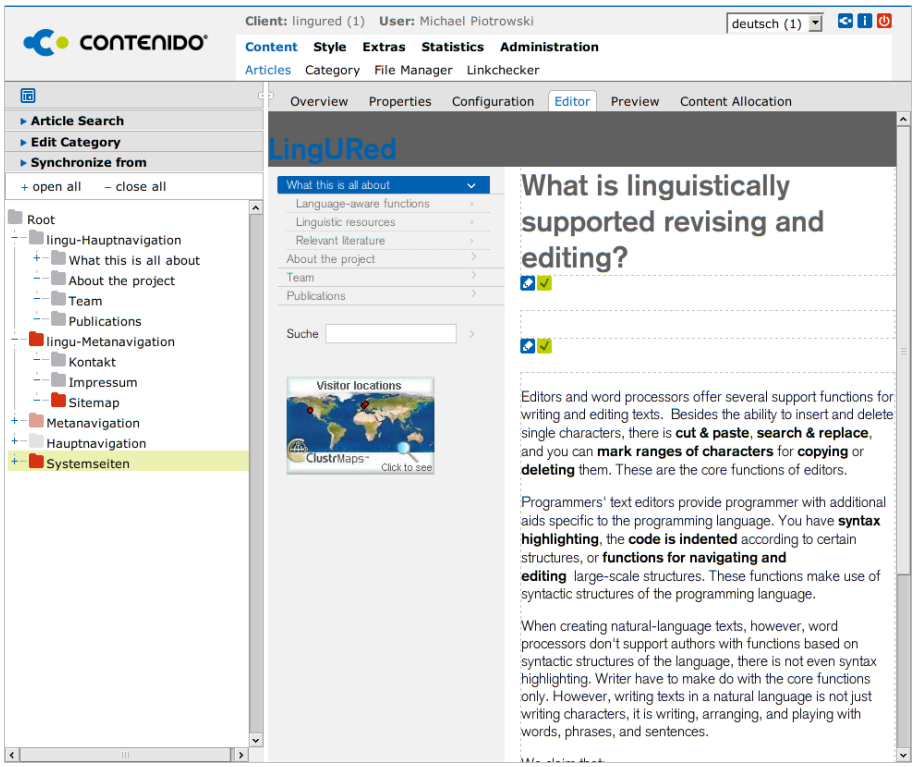


Figure 2.4: Screenshot of the Contenido user interface, showing a page tree on the left and a template-based content editor on the right, a layout typical for many CMSs.

2.5.1 E-Learning Platforms

There are hundreds of e-learning platforms today, both closed-source commercial products and open-source systems. The report of Baumgartner et al. [13] reflects the situation in 2002 but still gives a good overview of the number and diversity of systems available. Several academic institutions have also published their evaluation reports [e.g., 116, 28, 43].

Some of today's most popular platforms are the commercial systems Blackboard, Clix, Desire2Learn and WebCT[30], and the open-source platforms Ilias, Moodle, OLAT, and Sakai.

When reviewing e-learning platforms, there are, as for any software, three main views to consider:

1. The end user's view, i.e., the view of instructors and students using the platform. Both user groups have a number of basic and advanced requirements, which a platform must fulfill to be usable. The exact requirements partly depend on various external factors (e.g., the type of learning institution and the subjects) and also evolve over time.

2. The administrator's view, i.e., the view of the persons who install, administer, and maintain the platform, so that it can be used by end users.

3. The implementation view, i.e., aspects of the implementation, such as the programming language, architecture, and technologies used, as well as the license under which the system is dis-

30. Blackboard, Inc. acquired WebCT, Inc. in 2005; WebCT is now marketed under the name of Blackboard Learning System – Vista Enterprise License.

tributed. These aspects concern not only developers, but also affect end users and administrators, since they may, for example, influence the performance of the platform or the availability of certain features.

End users of e-learning platforms require support for the six activities listed in the definition in section 2.2.3: Creation, organization, delivery, communication, collaboration, and assessment. While details differ, all of the major e-learning platforms—which includes those listed above—support each of these activities to some degree. E-learning platforms thus can no longer be evaluated for their appropriateness in a certain environment by simply comparing feature lists. Monnard and Brugger note that "every product does have its specific strengths, where it outdoes most of its competitors, whereas it is lacking in other aspects," so that direct comparison is "nearly impossible." [116, p. 2]

From the administrator's view, most e-learning platforms require a database management system (DBMS); typical requirements are MySQL or PostgreSQL for open-source systems and Oracle or Microsoft SQL Server for commercial systems. For user authentication, most e-learning platforms can be integrated with directory services such as LDAP [189] or Shibboleth [118].

On the implementation level, current e-learning platforms use a wide variety of programming languages and architectures. Many systems are written in PHP (e.g., Ilias and Moodle), some in Java (e.g., OLAT and Sakai), others are based on Microsoft .NET (e.g., IntraLearn). Blackboard is written in a mix of Perl and Java. With respect to licensing, e-learning platforms can be separated into open-source and closed-source systems; the former (including Ilias, Moodle, OLAT, and Sakai) are usually developed by communities, released under the GNU Public License (GPL) or a similar license, and available for free, whereas the latter are typically developed and distributed by commercial vendors. Vendors usually do not make the source code of their products available and use various license agreements; cost is often dependent on the number of users.

Evaluations and experience show that once end users have adopted a platform, they tend to request additional features based on their practical use of the system. As can be seen in the release notes of e-learning platforms, these additional features frequently are not directly learning-related and already exist outside of e-learning platforms, such as calendars, chats, wikis, or social bookmarking.

Due to the license costs associated with commercial systems, universities are increasingly migrating toward open-source platforms (see section 4.5.2). Open-source software also mitigates the risk of vendor lock-in and allows universities to adapt the software to their special needs without having to develop a complete system from scratch; typical requirements are interfaces to campus management systems (such as HISInOne or SAP Campus Management) or alternative authentication solutions. A modular system architecture is thus desirable both with respect to administration and for the development of extensions and custom add-ons.

We will now briefly discuss and compare two e-learning platforms, Moodle and OLAT, which can be considered somewhat representative for the current state of the art.[31] Both platforms are in some respects similar (and thus comparable), but differ widely in other respects. Moodle and OLAT are open-source software; since universities are increasingly moving toward open-source platforms we do not consider commercial platforms.

Moodle

Moodle development was started in 1999 by Martin Dougiamas, who was working as a WebCT administrator at Curtin University of Technology, Perth, Australia, out of dissatisfaction with WebCT.[32] Version 1.0 was released in 2002. Moodle is licensed under the GNU Public License (GPL).

Moodle uses a traditional architecture based on the PHP programming language, the Apache Web server, and the MySQL database management system.[33] Even though it is possible to program in an object-oriented style with PHP, the design of Moodle is purposely *not* object-oriented (cf. footnote 32):

> Early on I made the decision to avoid using a class-oriented design – again, to keep it simple to understand for novices. Code reuse is instead achieved by libraries of clearly-named functions and consistent layout of script files. (`http://docs.moodle.org/en/Moodle_architecture` (accessed 2008-09-10))

Moodle is now being developed by a large community of developers. Moodle has a significant user base; as of October 2008, the Moodle Web site[34] speaks of over 50,000 registered sites. Even though it is clear that not all of them are actually in productive use, the number at least indicates an enormous interest in Moodle; equally impressive is the growth from about 15,000 registered sites in 2006.

From a user's perspective, most of the activities in Moodle take place inside *courses*, i.e., there are few functions that can be used outside the context of a specific course. Courses (see figure 2.5) themselves have a flat, non-hierarchical structure: Course authors can only choose from predefined course "formats," namely "Social" (oriented around one main forum), "Topics" (organized into topic sections, with each topic section consisting of activities), "Weekly" (organized week by week, with start and end dates, and each week consisting of activities);

31. The information used in the following discussion of Moodle and OLAT comes from a number of online sources, primarily the project Web sites `http://moodle.org/` (accessed 2008-10-17) and `http://www.olat.org/` (accessed 2008-10-17) and the documentation of the systems, the presentation *OLAT-Moodle comparison* by Florian Gnägi (July 2006, `http://www.frentix.com/de/olat/docu/OLAT-MOODLE-en.pdf` (accessed 2008-10-17)), and Bizoňová and Ranc [20], and the author's personal experience with OLAT at the University of Zurich.

32. Moodle stands for "Modular Object-Oriented Dynamic Learning Environment"; originally the "M" stood for "Martin's."

33. Other Web servers and DBMSs are also supported, but the combination of PHP, Apache, and MySQL—typically on a Linux platform—is the most common setup (often referred to as "the LAMP architecture").

34. See `http://moodle.org/stats` (accessed 2008-10-17).

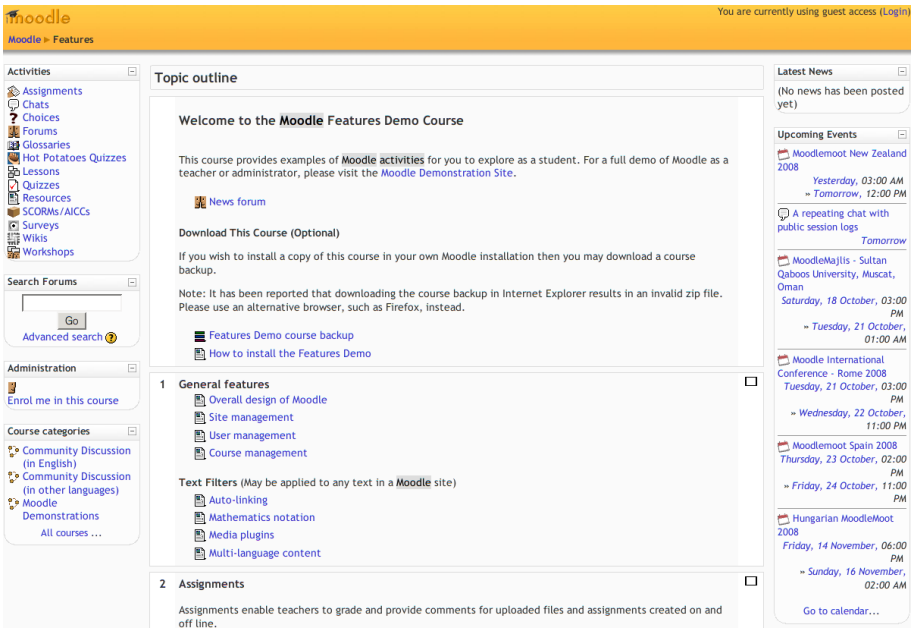


Figure 2.5: Screenshot of the Moodle "Features" demo course at `http://moodle.org/course/view.php?id=34` *(accessed 2008-10-09).*

activities is the Moodle term for functions such as assignments, chats, forums, quizzes, wikis, etc.

OLAT

Development of OLAT[35] also started in 1999, at the University of Zurich (Switzerland). The first version, developed by three student tutors[36] to help tutors cope with increasing numbers of students in computer science, was put into production in 2000. Like Moodle, the system was written in PHP and based on a similar architecture. This version was awarded the Medida-Prix[37] in 2000.

OLAT was quickly picked up by other departments at the University of Zurich and hosted centrally by the IT services. With increasing numbers of users, the developers realized that the existing architecture did not offer the modularity needed to allow the clean implementation of requested features and that it would not scale as required. A complete redesign, based on the experience made with the first version, and a complete reimplementation in Java was thus undertaken, leading to version 3.0 in 2004, which was released as open source software under the Apache Open-Source License. Also in 2004, the University of Zurich decided to use OLAT as their "strategic learning management system."

The redesign separated the infrastructure from the e-learning "business logic": OLAT 3.0 and subsequent releases (currently version 6.x) are based on a component-based architecture, i.e., OLAT is actually an application consisting of several components on top of a general-purpose framework. Even though it is not advertised as such, the framework can be used to build other Web applications.[38] Due to the experience

35. OLAT stands for "Online Learning And Training."
36. Florian Gnägi, Sabina Jeger, and Franziska Schneider.
37. The Medida-Prix is a trinational (Germany, Austria, Switzerland) competitive award (endowed with €100,000 of prize money total) for e-learning projects organized by the Gesellschaft für Medien in der Wissenschaft (GMW).
38. The framework is actually used for other applications.

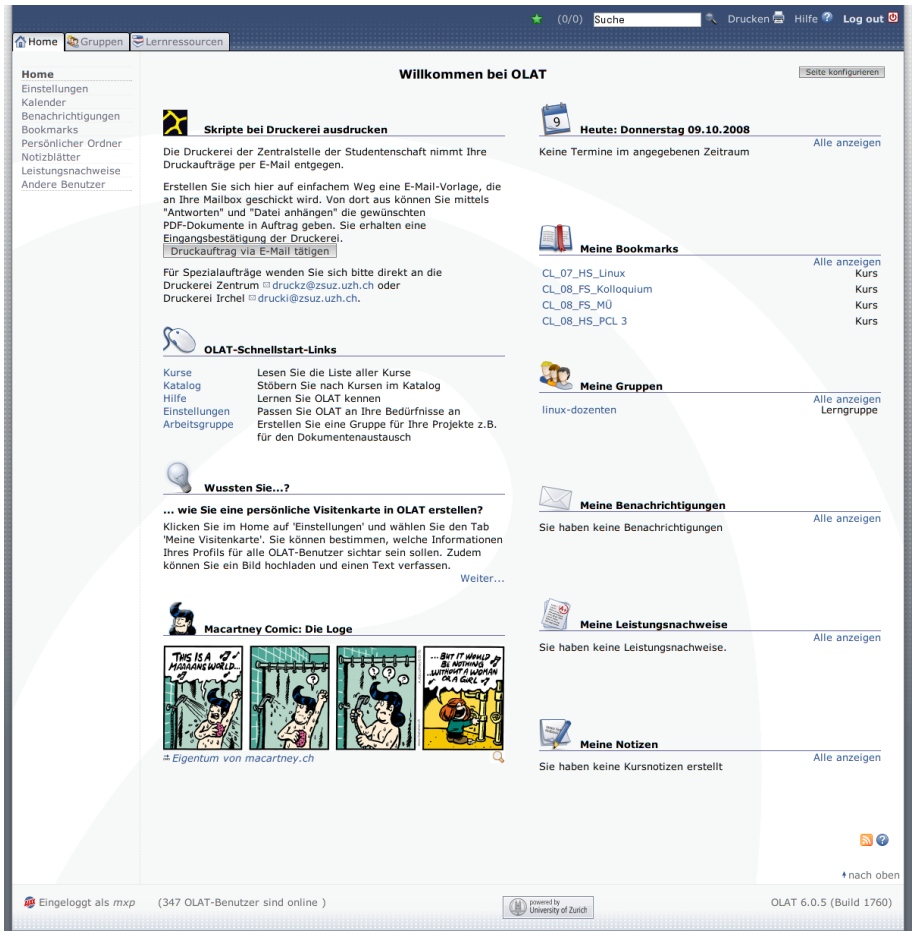


Figure 2.6: OLAT home page at the University of Zurich `http://www.olat.uzh.ch/` *(accessed 2008-10-17).*

with large numbers of concurrent users in the PHP-based version, the reimplementation also focused on scalability.

From the end user's (i.e., the instructor's and student's) point of view, OLAT differs from Moodle in many respects (see figure 2.6 for a screenshot). OLAT provides instructors and students with numerous tools that can be used outside of courses, adding groupware capabilities (such as synchronous and asynchronous communication, file sharing, and instant messaging) to the e-learning platform.

Courses in OLAT can have arbitrary hierarchical structures and there are no prescribed structures; instead, sequencing rules can be defined, if desired.

According to the Web site, there are about 150 OLAT installations running worldwide as of October 2008. The main OLAT installation is located at the University of Zurich (maintained by the Multimedia & E-Learning Services group); this installation is also used by other Swiss universities, such as the University of Basel, the University of Bern, the University of Lucerne, and the Swiss Federal Institutes of Technology in Zurich and Lausanne. It currently serves over 40,000 users, with about 400 to 700 concurrent users on average.

OLAT development is centered at the University of Zurich; contributions from outside developers have to be approved and integrated by the development team in Zurich.

Comparison

Moodle and OLAT are two e-learning platforms that represent the current state of the art. We will briefly compare some aspect of these two platforms to highlight some of the differences that can be found:

- In Moodle, most activities take place inside courses. OLAT, on the other hand, provides groupware-like functionality that can also be used outside of courses.
- Moodle requires instructors to choose from three predefined course structures. OLAT allows instructures to define their own structures and sequencing rules (which can be saved as "course templates.")
- Moodle advertises that it is "designed using sound pedagogical principles"[39], whereas OLAT emphasizes that it is pedagogically neutral.
- There are no official numbers on the number of concurrent users supported by Moodle; the Moodle installation documentation[40] notes that as a "general rule of thumb" Moodle supports 50 concurrent users for every 1 GB of RAM. The OLAT server at the University of Zurich serves a maximum of 900 concurrent users with 8 GB of RAM. A further bottleneck of Moodle is that, due to its architecture, data cannot be held in memory but must always be fetched from the DBMS or the file system.
- Moodle has a large and active developer community; however, there is only a low degree of central control, so that the quality of Moodle modules varies. OLAT is open-source, but with a high degree of central control: All contributions to OLAT must be approved by the developer team in Zurich. While this ensures good code quality, it slows down development and impedes the forming of a community.
- Both systems have the same age. However, while Moodle is still based on its original 1999 architecture, OLAT was radically redesigned in response to problems experienced with its original architecture, which was similar to that of Moodle.[41]

2.5.2 Content Management Systems

Many Web content management systems are available: *The CMS Matrix*[42], a Web site collecting information on CMSs, lists almost 1000 systems[43] There are both numerous commercial, closed-source offerings from vendors such as EMC, IBM, Microsoft, OpenText, Oracle, and Vignette, and many popular open-source systems, e.g., Contenido, Drupal, Jaws, Apache Lenya, Nuxeo CPS, OpenCms, Plone, Silva, and TYPO3.

39. `http://docs.moodle.org/en/Philosophy` (accessed 2008-10-17)
40. `http://docs.moodle.org/en/Installing_Moodle` (accessed 2008-10-17)
41. Interestingly, the current OLAT architecture is based on design principles similar to those of the eduComponents (see chapter 3).
42. `http://cmsmatrix.org/` (accessed 2008-10-17)
43. On September 9, 2008, *The CMS Matrix* listed 961 systems.

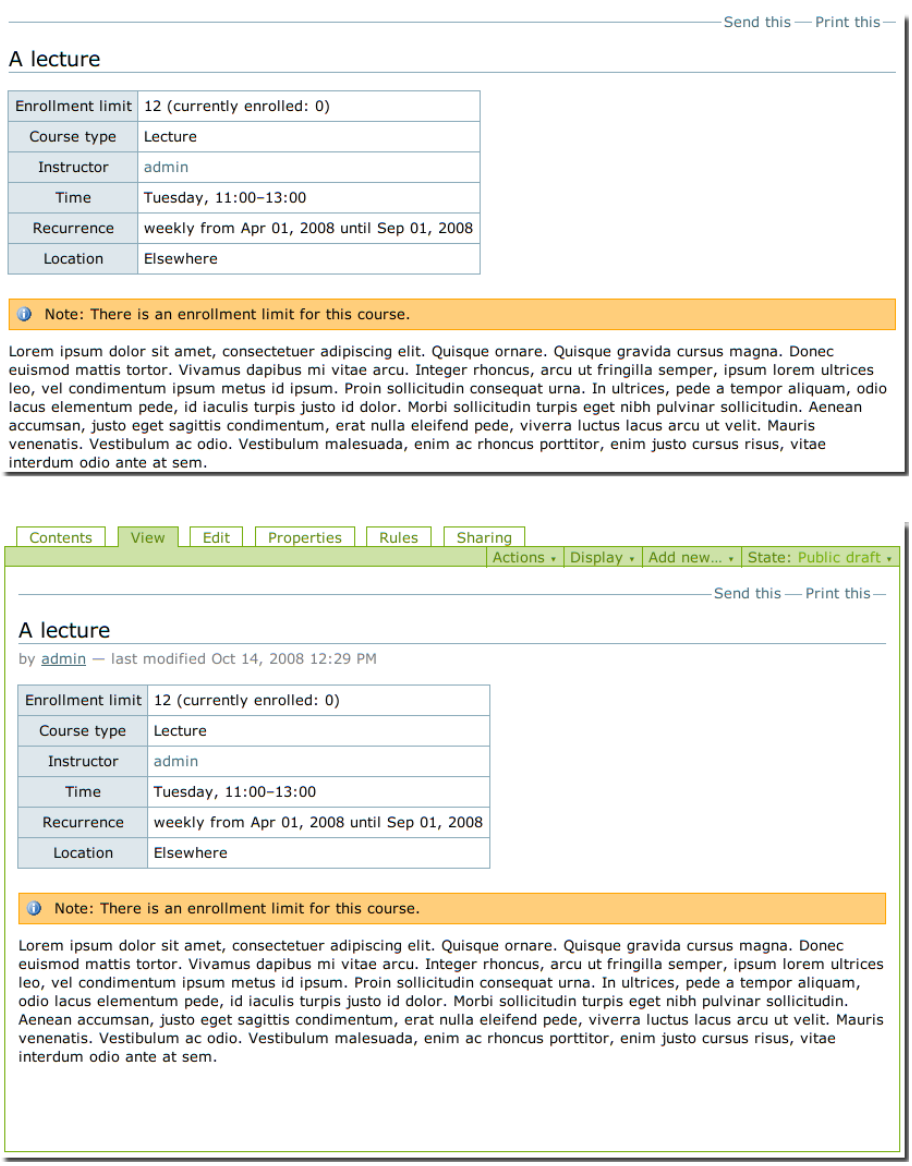


Figure 2.7: The user interface of Plone does not separate the authoring interface from the published Web site, as most other CMSs do. Instead, the available tools and functions depend on the user's permissions. The upper screenshot shows a page as it appears to an anonymous user; the lower screenshot shows what an authenticated user sees, who is permitted to edit the page.

As for e-learning platforms, there are three main views to consider: (1) The end users' view—in the case of a CMS these are primarily the content authors, as the presentation readers see does for the most part not depend on the CMS—, (2) the administrators' view, and (3) the implementation view. For the latter two the same points apply as for e-learning platforms (see p. 48).

End users of content management systems require support for creation, organization, and publication of content. Most CMSs strictly separate the views of content authors and readers: Authors access the CMS using a Web interface separate from the published Web site, which provides access to tools for creating and maintaining content. Figure 2.4 on page 48 shows a typical user interface for authors in a CMS.

Plone—which serves as the basis for the eduComponents approach described in this dissertation (see section 3)—differs in this respect from most other CMSs in that there is no separate authoring interface. Instead, the available tools and functions depend on the user's permissions, see figure 2.7. This approach allows authors to navigate the

Web site as it appears to readers without having to switch to a separate mode to make edits.

With the exception of smaller, "personal" systems, CMSs are expected to provide support for publishing workflows, including approval mechanisms, and advanced publication features, such as timed publication.

In contrast to e-learning platforms, which usually implement fixed structures and navigation elements, the appearance and structure of CMS-managed content is largely independent of the actual CMS used. While every CMS provides some default elements, which are often used and thus contribute to a specific "look and feel" of Web sites managed by that CMS, most visual and structural designs can be implemented using any CMS.

From the administrator's perspective, most content management systems require a Web server and a separate database management system (DBMS); typical requirements are MySQL or PostgreSQL for open-source systems and Oracle or Microsoft SQL Server for commercial systems. Some systems, such as Plone, come with an integrated Web server and/or DBMS. With the exception of "personal" CMSs, most CMSs can be integrated with directory services such as LDAP [189] or Shibboleth [118] for user authentication.

Current CMSs are implemented in a wide variety of programming languages and are based on diverse architectures. Many systems are written in PHP (e.g., TYPO3 or Drupal), some in Java (e.g., Apache Lenya), Perl (e.g., Bricolage or Imperia), or Python (e.g., Plone or Silva), others are based on Microsoft .NET (e.g., DotNetNuke). Different CMSs target different needs and thus differ with respect to performance and scalability.

With respect to licensing, content management systems can be divided into open-source and closed-source systems. Due to license costs and restrictions associated with commercial CMSs, universities—and especially smaller units, such as departments and institutes—are mostly using open-source systems. As in the case of e-learning platforms, open-source software reduces the risk of vendor lock-in and allows universities and departments to adapt the software to their individual requirements without having to develop a complete system from scratch.

Most CMSs are extensible with add-ons or modules and can thus offer functionality beyond standard content management, such as blogs, chats, forums, wikis, photo galleries, calendars, or e-commerce facilities (shopping carts, inventory management, interfaces to credit card companies).

Regarding the actual delivery of content, there are two fundamental approaches: Offline processing (or staging), and online (or live) processing. In the first case, all content to be published is assembled into static pages and exported to a Web server, whereas in the second case, pages are only created when requested by a reader, on the fly. Online Web CMSs are the dominant variety today, as servers are fast enough for on-the-fly assembly of pages, and this approach enables better personalization and ensures that all content is always current.

Bergstedt et al. [16] list a number of features a "perfect CMS" should, in their opinion, have. Their list provides a brief overview of the functionality and features currently expected from content mangement systems:[44]

- Separation of content, structure, and presentation to enable "single source, multiple media" publishing
- Asset management
- Version management for content and assets
- Workflow management and support for editorial processes
- Users and role management
- Ability to import and export content and assets
- Publication
- Individualization of the content presentation
- Content syndication
- Content archiving
- Extensibility through scripts or modules

To summarize this section, current state-of-the-art content management systems can be said to provide the listed features—however, they differ widely in their implementation, performance, and usability.

Many systems commonly referred to as *content management systems* implement numerous features beyond traditional content management functionality, especially communication functions, such as forums or chats, or calendars, and syndication and personalization functionality. Sometimes terms like *portal server* or *portal platform* are used to describe such systems [see 181].

We also note that these systems, whatever they are called, effectively support five of the six activities required for e-learning platforms (see p. 42), with the exception of assessment.

2.6 Summary

In this chapter we have laid a foundation for the subsequent chapters by examining central terms—*e-learning*, *e-learning platform*, and *content management system*—and their definitions. We have found that these terms are generally not well-defined and have thus devised our own practical definitions for this dissertation. Since this dissertation is only concerned with e-learning in academic environments, we have also outlined the most significant differences between corporate and academic e-learning.

44. The list given here is a revised version of the one given in [16], with spelling and language errors corrected, duplicates removed, and reordered.

We have given a brief overview of the PLATO system to give a historical perspective to e-learning platforms and we have described the state of the art of e-learning platforms and content management systems.

Most importantly, we have discussed the need for specialized e-learning platforms on the basis of our definition of an e-learning platform as system supporting six activities in an educational context, viz. creation, organization, delivery, communication, collaboration, and assessment. Our conclusion is that modern Web content management systems support five of these activities, i.e., with the exception of assessment few of these activities are educational *per se*.

In the next chapter we will describe the eduComponents approach, which is based on this realization.

3 The eduComponents Approach

In the previous chapter, we have defined e-learning platforms as systems that support six activities in an educational context: Creation, organization, delivery, communication, collaboration, and assessment (see p. 42). Furthermore, we have concluded that modern Web content management systems support five of these activities. This means that few of these activities are educational *per se*—the one exception is assessment.

So, to put it bluntly, one could say

e-learning = Web + assessment.

We would further argue that the "Web" part (ie., creation, organization, and delivery of content, communication, and collaboration) are better supported by extensible content management systems (or "portal servers"), in the sense that CMSs are *more general*, since they are not only designed for one specific application scenario, thus *more flexible*, and *more mature*, as they are used by more people in more—and more diverse—environments.

Our conclusion—and one of our theses—is thus that it would be advantageous to separate the application-specific functionality (or "business logic") from the infrastructure and the presentation, and to build e-learning platforms on the proven infrastructure of content management systems instead of reimplementing their functionality.

However, obviously not *every* CMS constitutes an appropriate basis for building an e-learning platform. To provide a viable basis with respect to software engineering considerations, it should be characterized by a suitable architecture. A suitable architecture will—at least—allow developers to implement extensions (in our case, those needed for e-learning, i.e. assessment functionality) without having to modify the base system. To allow system administrators to adapt the platform to the needs of their users, the CMS should not just be "modular" in a

general sense, but individual modules should be deployable individually. This requirement implies a *component-based architecture* [173] or possibly a *service-oriented architecture* [131].

The approach presented in this dissertation uses a component-based architecture and consists of a collection of e-learning components, named eduComponents, for the Plone content management system. The eduComponents approach is characterized by the following three principles:

- **Component-based architecture**
  Institutions or individuals operating and administering an e-learning platform should only be required to install those components they actually need, and users (students and instructors) should not have to concern themselves with functionality they do not need. If desired, it should be possible to combine these components with others, including components from third parties.

- **Uniform content representation**
  All types of content should be represented in a uniform way to offer a single, consistent interface both to users and developers. This interface should be based on the model of documents; we refer to this approach as *document orientation*.

- **Proven infrastructure**
  A large part of e-learning effectively consists of content management. Instead of reimplementing the basic content management functionality, as conventional e-learning platforms do, a content management system should be used as the foundation.

The following sections describe the design and implementation of the eduComponents as a whole and the individual components. The eduComponents were conceived, designed, and implemented collaboratively by the author and Mario Amelung while working in the WDOK research group at Otto von Guericke University Magdeburg. ECQuiz was designed by the author, and implemented by Wolfram Fenske under the supervision of the author. The Quick Edit functionality of ECQuiz was designed by the author, and implemented by Sascha Peilicke and Wolfram Fenske under the supervision of the author. ECSpooler was implemented by Mario Amelung, and by Wolfram Fenske under the supervision of Mario Amelung, with contributions by the author. As the first users of the eduComponents products, the members of the WDOK research group tested pre-release versions and contributed feedback and bug reports.

3.1 Design and Implementation

3.1.1 Design Considerations

The eduComponents are a new approach to the architecture of an e-learning plattform. The design of the eduComponents tries to overcome the technical problems with conventional e-learning platforms,

as described in the previous chapters, and to enable new approaches for deployment.

The design of the eduComponents was guided by the the following three questions:

a) Is it possible to use a shared platform for both e-learning and other Web content?

b) Is it possible to use a component-based architecture—enabling the tactical combination of components according to concrete requirements—instead of the typical integrated systems?

c) Is it possible to use a uniform, transparent representation for all types of content?

Question (a) is motivated by the observation that a large percentage of e-learning is actually document management and that five of the six e-learning activities are supported by content management systems.

The motivation for question (b) is the size and inflexibility of conventional e-learning platforms, and the consequential deployment, maintenance, and usage issues described above.

Question (c) can be considered as an extension of a, i.e., if the functionality of e-learning platforms can be abstracted from their educational context to arrive at more general functions, can the different types of objects involved also be reduced to a smaller number of generic classes?

The developers of OLAT recognized the need to separate e-learning functionality from the content management infrastructure (and from the presentation) and developed a *three tier architecture* [e.g., 191] for OLAT 3, based on a general framework (see section 2.5.1). They then developed—from scratch—a general framework, on top of which they built the e-learning platform OLAT. The framework is not specific to e-learning and can also be used for other applications, e.g., a content management system.

However, developing a content management infrastructure requires a significant investment. When the Knowledge-Based Systems and Document Processing Research Group (WDOK) at Otto von Guericke University Magdeburg needed an e-learning platform in 2003, we could not afford to implement a comprehensive infrastructure from scratch. Instead, based on the considerations outlined above, we were looking for a suitable content management system, so that we would only need to implement the missing, e-learning-specific functionality.

The design decisions taken in response to the first two questions are thus: Instead of re-implementing CMS functionality—as conventional e-learning platforms do—, the eduComponents rely on a general-purpose CMS to provide (1) a rich development environment and a basis for deploying components, and (2) a reliable implementation of basic document management functions. Furthermore, the eduComponents approach relies on a component-based architecture, which makes it possible to install only the required components and to combine components from different sources. The third point is to use a uniform

content representation, in our case, documents are the central objects, and tests are documents just like lecture notes.

3.1.2 Implementation

Zope and Plone

For the actual implementation of the eduComponents we selected the Plone[1] content management system as the basis for our components. Plone is built on top of the Zope[2] Web application framework.

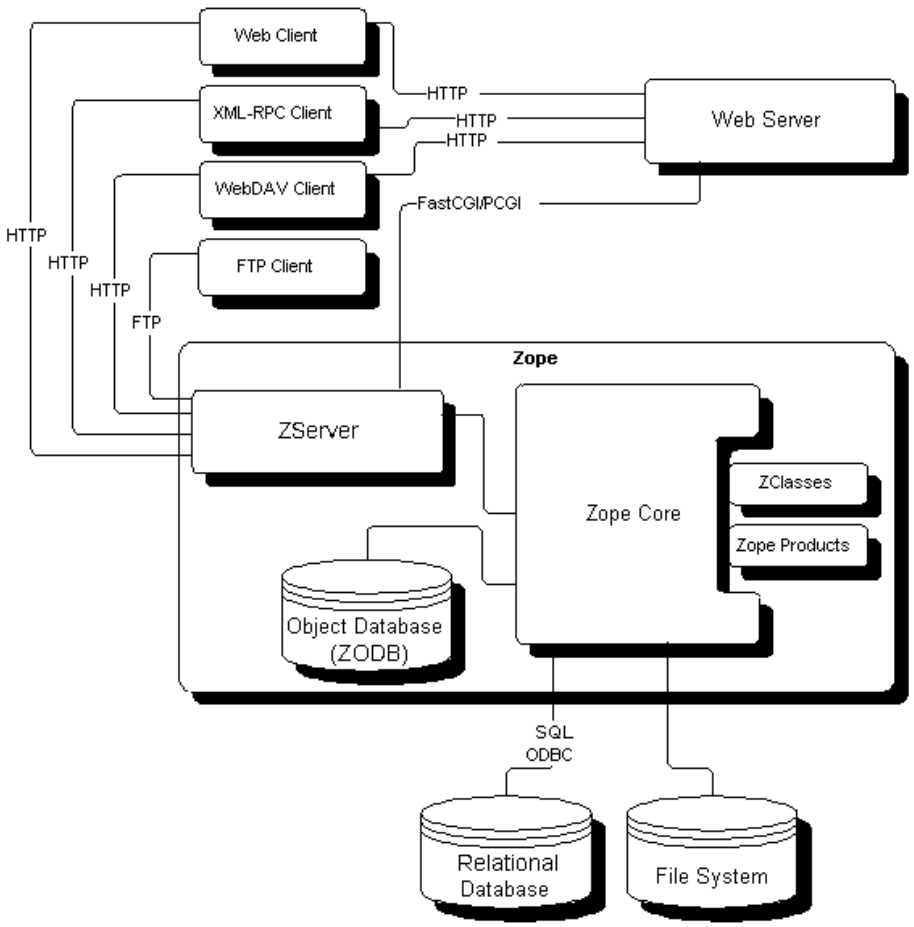


*Figure 3.1: Diagrammatic overview of the Zope architecture.**

Zope provides a platform for for building Web applications such as content management systems, intranets, portals, or other custom applications. Zope is based on a transactional object database (ZODB); it also includes an integrated Web server (ZServer), a search engine (ZCatalog), and a workflow engine. It can connect to all major relational database management systems, interoperate with other Web servers (such as Apache), and supports further network protocols such as FTP, WebDAV, and XML-RPC (see figure 3.1).

The component-based architecture of Zope (see below) encourages the development of extension modules (so-called *products*); over 300 community-developed products are currently available from the Zope

1. `http://plone.org/` (accessed 2008-10-17)
2. `http://zope.org/` (accessed 2008-10-17)

* Taken from `http://www.zope.org/WhatIsZope/ZopeArchitecture` (accessed 2008-10-17))

Web site, providing special functionality for e-commerce applications, for integration and interoperability with other systems, or for user management, to name just a few areas. In brief, Zope provides the complete infrastructure necessary for developing Web-based applications.

Plone is a content management system based on Zope. Plone offers all standard CMS functionality (see sections 2.4 and 2.5.2), general-purpose content types like documents and folders, and more specialized ones such as hyperlinks, images, calendar events, or news items. Plone shares the component-based architecture with Zope, so many other content types can be added via products. Currently over 900 products developed by the community are available from the Plone Web site, providing additional or enhanced functionality in areas such as layout and presentation, versioning, communication, media management, workflow, calendaring, or polls and surveys. All content objects, whether of standard or add-on types, have associated metadata; the default set of metadata is based on the Dublin Core Metadata Element Set[3]. Add-on products are available which extend the standard full-text search with ontology-based search options. Content objects can also have other associated information, including discussion threads.

Plone is highly extensible and adaptable. Simple custom content types and associated views, security policies, and workflows can be defined with only minimal programming skills through a Web interface. More complex types obviously require corresponding programming skills; the actual amount of programming can, however, be drastically reduced by using ArchGenXML, a code generation tool, which allows developers to model new content types in UML and to specify the associated workflows, and then have all code necessary for deployment as a Plone product generated automatically from the specification.

Both Zope and Plone are open source and licensed under the GNU Public License. Zope and Plone are written in Python[4], a widely-used and highly portable open-source object-oriented programming language. This enables Zope and Plone to run on a very large number of platforms, including practically all UNIX and UNIX-like systems (e.g., Linux, BSD variants, Mac OS X), as well as Microsoft Windows. Zope and Plone are supported by active, world-wide communities of thousands of individual developers and hundreds of companies offering Zope- and Plone-based solutions.

After evaluating a number of other systems, we chose Plone for its portability, its conformance to Web standards—including accessibility standards[5]—, its workflow capabilities, and, most importantly, for its architecture, which enables the fast development and deployment of components (products). Plone products can make use of the basic CMS functionality and interact with other products through well-defined

3. `http://dublincore.org/` (accessed 2009-01-10)
4. `http://python.org/` (accessed 2008-10-17)
5. *Accessibility* refers to the usability of software or Web sites by persons with visual or motor impairments. According to the Plone documentation, "Plone was the first Content Management System in the world to be compliant with the WAI-AA [W3C Web Content Accessibility Guidelines, conformance level Double-A] and US Section 508 accessibility standards" (`http://plone.org/products/plone/features/3.0/existing-features/accessibility-compliant` (accessed 2009-01-10))

interfaces; a large number of products for a wide variety of applications is readily available.

Building on portable open-source software also avoids license fees and vendor lock-in, and has other advantages, including complete control over the software and the data, the possibility of adapting it to one's specific needs and a large developer community.

Component-Based Architecture

As mentioned above, Zope implements a *component-based architecture*. The Zope Core provides the infrastructure—or the "foundational functionality" [173, p. 12]—on which Zope components—called *products* in Zope terminology—can be deployed (see figure 3.1).

As the name implies, a component-based architecture relies on *software components*. The most widely accepted definition of *software components* is that by Szyperski et al.:

> A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties. [173, p. 41]

This definition describes three essential characteristics of software components (see Szyperski et al. [173], p. 36):

1. A component is a "unit of independent deployment," encapsulating its consitutent features.
2. A component is a "unit of third-party composition," i.e., it is composable with other components by a party who does not have access to the construction details of the components involved; this requires that a component encapsulates its implementation and interacts with its environment by means of well-defined interfaces.
3. A component does not have any (externally) observable state.

Zope products generally fulfill these requirements and can thus be regarded as software components in the sense of the definition. However, as software packaging and distribution units, products may, for practical reasons, not directly correspond to components: They may contain more than one component, or a sub-component unit, such as an optional feature of another component, may be distributed as a separate product (see below for an example). This is, however, in line with Szyperski et al.'s view: They emphasize that in successful component frameworks, "components exist on a level of abstraction where they directly mean something to the deploying client" [173, p. 13] and note the example of Web browser plug-ins, where "the client gains some explicable, high-level feature and the plug-in itself is a user-installed and configured component" [173, p. 13]. This is contrasted with objects, which only have a meaning to programmers and composition or assembly of objects is only possible for programmers as well.

Internally, Zope products use an object-oriented design. In particular, Plone products (such as the eduComponents products) generally expose one (or sometimes more) classes that represent specific, user-visible *content types*.

Plone, then, is technically a collection of Zope products, i.e., software components, which are just distributed together for convenience, just like the eduComponents. However, a simplified view could be to regard collections of components as "layers," with the lower layers providing functionality used by the higher layers. Figure 3.2 is a graphical representation of this view, where Zope would constitute the lowest layer, followed by CMF, the Zope Content Management Framework[6], Plone, and finally Plone products, such as the eduComponents.

Another simplified view is to consider Plone as an application which can be enhanced by extension modules, providing, for example, new content objects, connectors to relational databases and other external data sources, or, as the eduComponents do, e-learning functionality. While this view is, strictly speaking, not correct, it can help to explain the architecture to non-technical users.

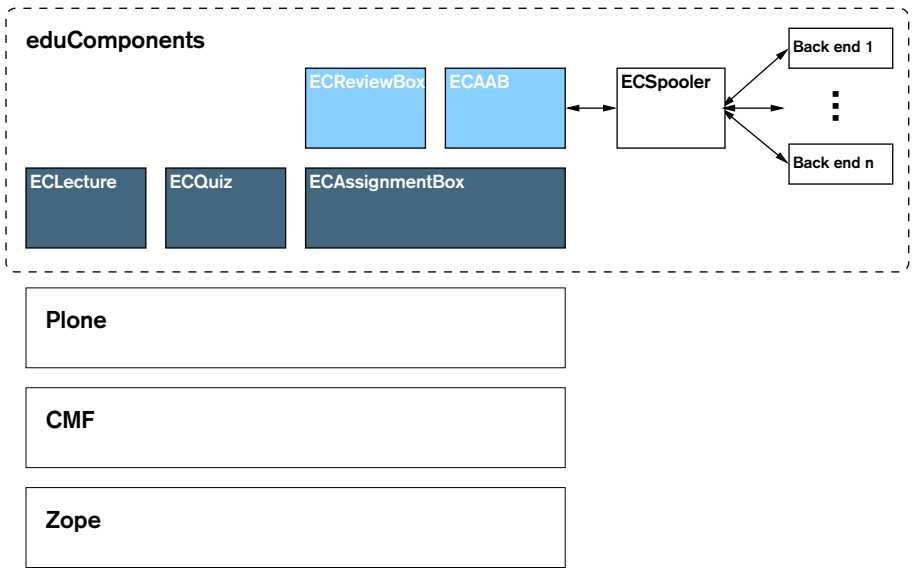


Figure 3.2: Schematic diagram of the eduComponents architectural, showing the layers on which the eduComponents Plone products (shaded boxes) are based, and the ECSpooler system, which is independent from Plone. The arrows indicate XML-RPC communication. The term eduComponents usually refers to all components inside the dashed box.

Note that there is not in all cases a one-to-one correspondence between the eduComponents products shown in figure 3.2 and software components in the sense of the above definition: ECAutoAssessmentBox and ECReviewBox (colored in a lighter shade in figure 3.2) are derived classes inheriting from ECAssignmentBox and are therefore not usable without ECAssignmentBox—obviously, using ECReviewBox without ECAssignmentBox would not make sense.

Thus, as they cannot be deployed independently of ECAssignmentBox, ECReviewBox and ECAutoAssessmentBox are not software components according to the definition. They could be regarded as sub-component units of ECAssignmentBox; they do have *some* component-like properties (such as being installable by non-programmers and pro-

6. CMF is a collection of components providing a platform for content management applications. CMF is also used by other CMSs based on Zope, e.g., Nuxeo CPS `http://www.cps-project.org/` (accessed 2008-10-22).

viding explicable, high-level features), but logically and technically depend on ECAssignmentBox.

3.2 Short Description of the Individual Components

This section gives a brief overview of the eduComponents products developed in the course of this dissertation. These products are:

ECLecture: Course information, registration, and resource management.

ECQuiz: Multiple-choice tests.

ECAssignmentBox: Assignments and submissions, with support for assessment, grading, and plagiarism detection.

ECAutoAssessmentBox: A variant of ECAssignmentBox offering automatic checking and assessment of assignments with immediate feedback.

ECReviewBox: A variant of ECAssignmentBox for peer review of submissions to ECAssignmentBox and ECAutoAssessmentBox assignments.

With respect to our definition of e-learning platforms (see p. 42), ECLecture can be said to support the organization activity, while the other products implement assessment activities.

The eduComponents products can be used separately or in combination, and they can also be combined with other Plone products to create tailor-made learning environments. For example, if a bibliography, a glossary, a wiki, or a discussion forum is needed, third-party Plone products can be added. Also, all components use a uniform content representation. All objects in Plone are documents (or folders containing documents) and can be manipulated in the same ways, regardless of whether the document is a multiple-choice test or an image. This ensures a consistent and easy-to-learn user interface.

Another common feature of the eduComponents products is that they are all fully internationalized, i.e., they can easily be adapted to different language environments. This is enabled by the use of the standard internationalization facilities of Zope: The products' user interface texts and messages are stored separately in a so-called *message catalog*; to localize a product for a new language environment, a translator only has to translate the texts in the message catalog. All eduComponents products are localized in English and German; ECQuiz is additionally localized in French, Italian, and Slovenian.[7]

The eduComponents products are licensed under the terms of the GNU Public License and are freely available as open-source software from the addresses given in the following descriptions.

7. The French translation was done by the author, the Italian and Slovenian translations were contributed by users from Italy and Slovenia.

3.2.1 ECLecture

ECLecture is a Plone product for managing lectures, seminars and other courses. Figure 3.3 shows a typical view of a course homepage realized with ECLecture.

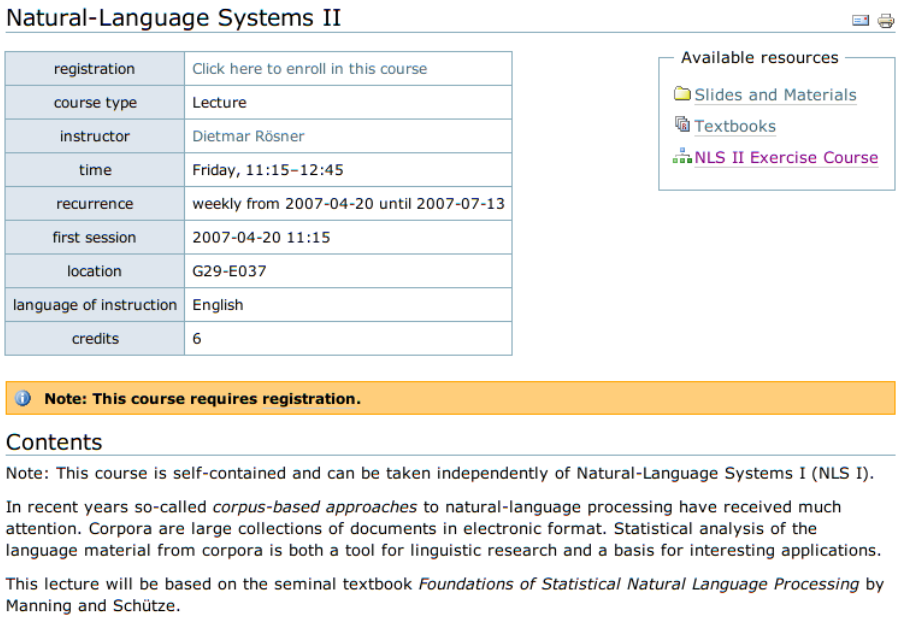


Figure 3.3: Example course homepage realized with ECLecture. To ensure readability, the site-wide navigation provided by Plone is not shown in this and the other screenshots

ECLecture objects group all course-related information, i.e., course metadata such as title, instructor, time, location, credits, etc., as well as course resources. ECLecture objects can thus serve as "portals" to all course-related materials like slides, exercises, tests, or reading lists. These materials are managed using the appropriate content types (e.g., ECAssignmentBox for assignments) and appear as resources to the course. Since an ECLecture object is a folder-like object, these resources can be stored inside of it, but they can also be stored somewhere else, including other Web servers.

ECLecture extends the organization and delivery functions of Plone by providing a structuring tool with e-learning-specific semantics.

ECLecture also handles registration for courses. Course enrollment is another example of how the eduComponents assign domain-specific semantics to basic Plone functionality: With regard to enrollment, a course corresponds to a user group; thus, registration for a course is realized by assigning Plone users to the group corresponding to the course.

ECLecture is available from `http://plone.org/products/eclecture` (accessed 2008-10-23).

3.2.2 ECQuiz

ECQuiz is the eduComponents product for multiple-choice tests. The product supports single-answer, multiple-answer, scale, and tutor-graded free-text questions. Related questions can be grouped into question groups, which are then treated as a unit. Questions and answers can be displayed in fixed or randomized order. It is also pos-

sible to present different randomly selected subsets of questions and answers to each student. Figure 3.4 shows a typical view of a test.

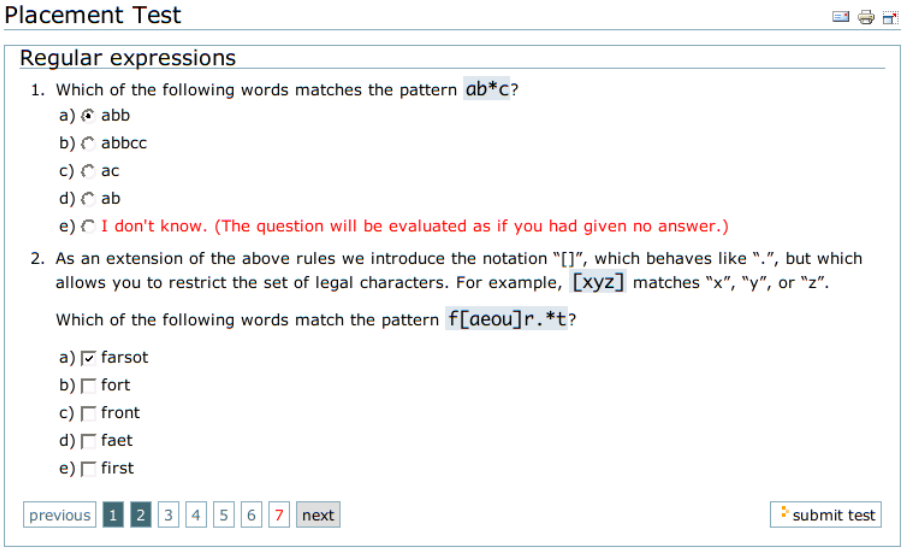


Figure 3.4: Screenshot of a multiple-choice test in ECQuiz.

ECQuiz offers different modes of operation for self-assessment tests and exams. Instructors can access detailed reports, providing an overview of the performance of all test takers. The reports can also be exported for further processing in a spreadsheet or statistics program.

More details are given in section 4.4.1. ECQuiz is available from `http://plone.org/products/ecquiz` (accessed 2008-10-23).

3.2.3 ECAssignmentBox

ECAssignmentBox implements the creation, submission, grading, and management of essay-like assignments. *Essay-like* means that the students' answers are more or less unrestricted, free-format text. If the file upload facility is used, students can submit any file, including non-text documents, such as audio, graphics, or specialized application files. ECAssignmentBox allows instructors to define time windows during which submissions are allowed. Figure 3.5 on the facing page shows an assignment box as it presents itself to students.

The assessment process is semi-automated, i.e., the assessment is done by the instructor, who is aided by the tool during the process of grading students' work and giving feedback. ECAssignmentBox provides the instructor with management, statistics, and analysis functions for the submissions in assignment boxes.

Furthermore, assignment boxes can be grouped together using a specialized folder type (ECFolder) to form "exercise sheets," courses, or similar units. ECFolder extends the statistics and analysis functions to all submissions in all contained assignment boxes.

The assessment of student submissions can be described as a process that starts with the submission of a student's answer and ends with the assignment of a grade by the instructor. ECAssignmentBox models this process using Plone's workflow facilities and defines a specialized workflow for student submissions.

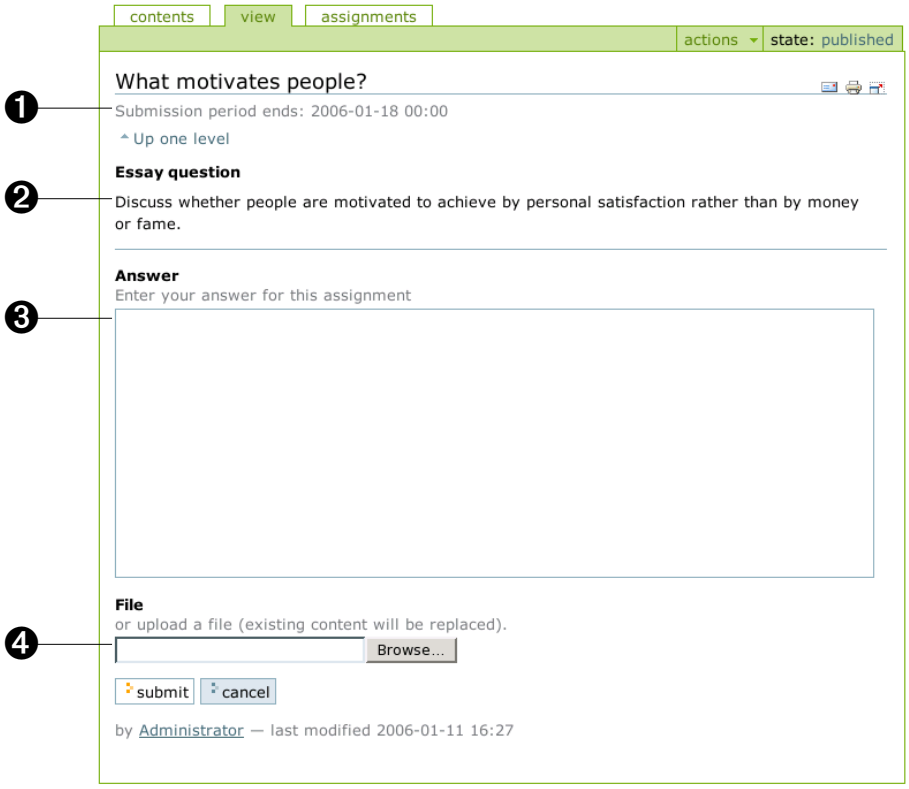


Figure 3.5: Student's view of an assignment box. At the top of the assignment box, students are informed about the submission period ❶. The main part consists of the assignment text ❷and the answer field ❸; alternatively, a file can be uploaded ❹

More details can be found in section 4.4.2. ECAssignmentBox is available from `http://plone.org/products/ecassignmentbox` (accessed 2008-10-23).

3.2.4 ECAutoAssessmentBox

ECAutoAssessmentBox is a Plone product derived from ECAssignmentBox, i.e., it inherits the same basic functionality as described in section 3.2.3 above. In addition, ECAutoAssessmentBox allows student submissions to be checked and evaluated automatically. ECAutoAssessmentBox acts as a front end to a Web service, ECSpooler, which handles the evaluation of submissions. This is especially interesting for programming assignments; however, with the appropriate backends, the system can also be used to check submissions in other formal notations or to analyze natural-language answers.

ECAutoAssessmentBox allows students to submit their solutions for programming assignments via the Web at any time during the submission period. Submitted programs are automatically checked and students get immediate feedback on whether their programs are syntactically correct and yield the expected results, as shown in figure 3.6 on the next page.

A more detailed description of ECAutoAssessmentBox follows in section 4.4.3. The product can be obtained from `http://plone.org/products/ecautoassessmentbox` (accessed 2008-10-23).

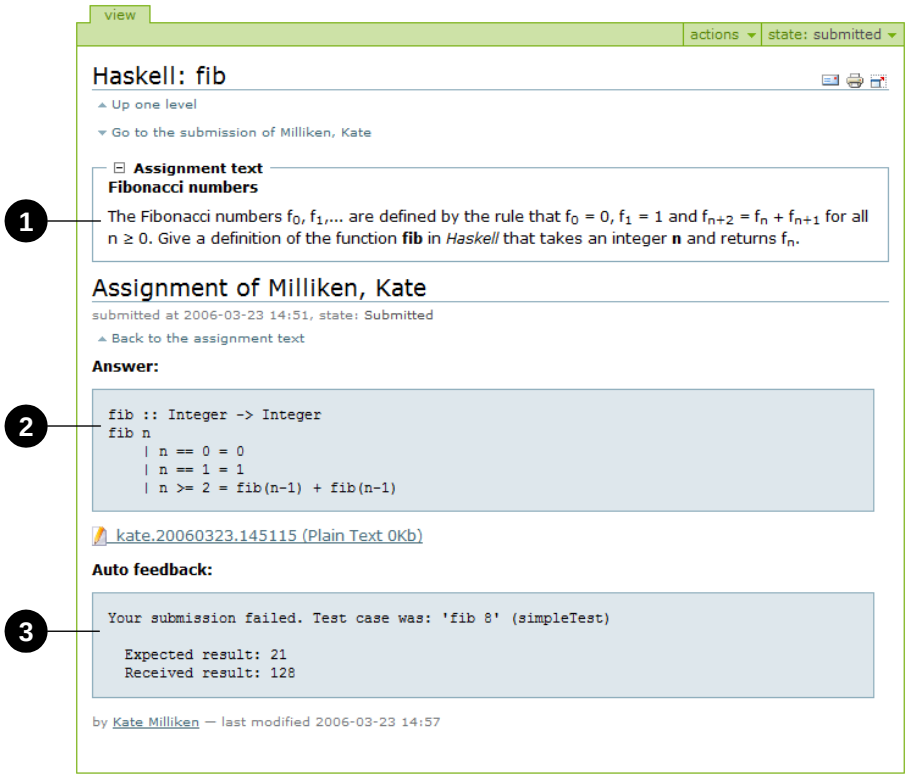


Figure 3.6: ECAutoAssessmentBox automatically tests submissions to programming exercises and immediately offers feedback. This screenshot shows the view students may get after the automatic testing of their programming assignment. ❶ is the assignment; ❷ is the student's submitted program, in this case an incorrect solution; ❸ is the automatic feedback, reporting an error, since the submitted solution does not yield the expected results.

3.2.5 ECReviewBox

ECReviewBox is essentially an add-on for ECAssignmentBox and ECAutoAssessmentBox, allowing instructors to create *peer review assignments* based on submissions to a previous assignment.

When using ECAssignmentBox or ECAutoAssessmentBox for online submissions to assignments, all submissions are available as electronic documents, so that they can easily be processed and distributed. ECReviewBox takes advantage of this and enables their use for peer reviews.

Peer review here means that students are given anonymized copies of other students' submissions and are asked to review them according to some set of criteria defined by the instructor (see figure 3.7 on the facing page).

More details on ECReviewBox are given in section 4.4.4. The product is available from `http://plone.org/products/ecreviewbox` (accessed 2008-10-23).

3.3 Related Work

In this section we will briefly review work related to the eduComponents. Related work can be classified into several areas: Research on architectures for e-learning platforms, actual systems based on designs similar or related to the one described in this dissertation, and other uses of Plone in e-learning.

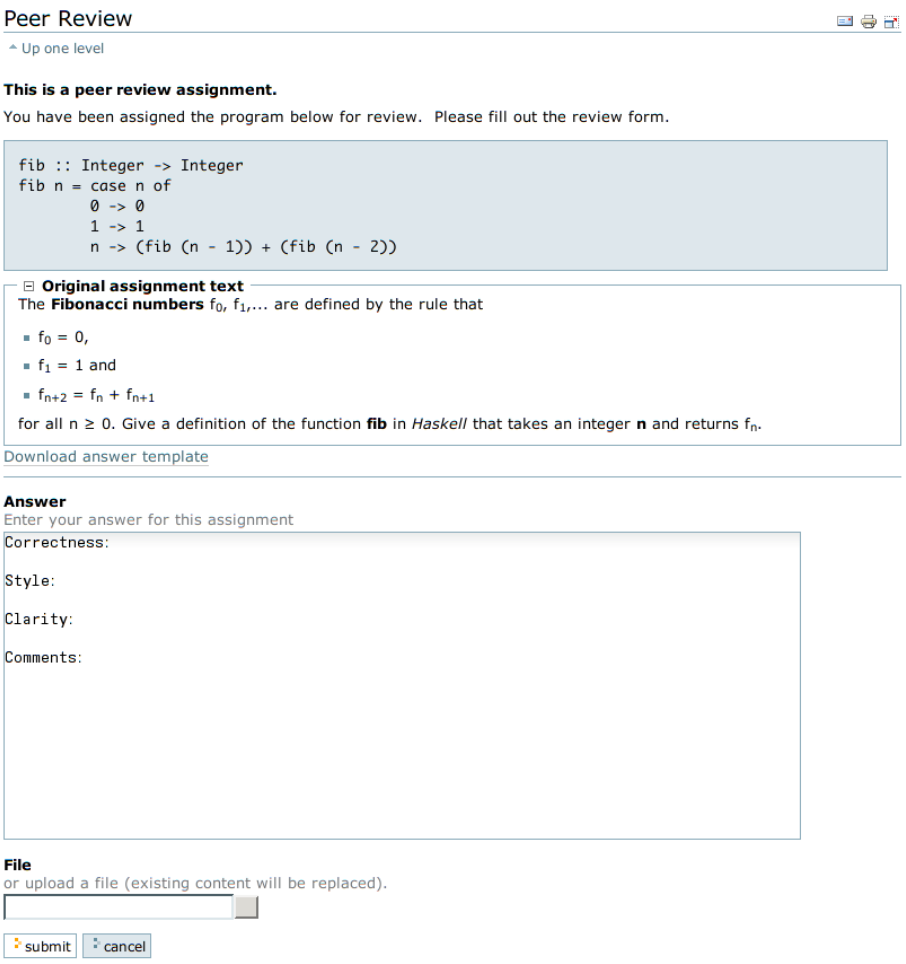


Figure 3.7: ECReviewBox allows instructors to create peer review assignments. In this example, the student is given another student's submission to a previous assignment for review.

3.3.1 Architecture

The eduComponents approach is marked by two essential characteristics: (1) The abstraction of functionality from the e-learning domain to more general functionality as found in CMSs, and (2) the component-based architecture, which allows to selectively combine e-learning functionality into a customized e-learning platform.

We have found almost no research related to the first point. There is a blog posting from 2004 by Morrison which asks some of the same questions that also prompted the development of the eduComponents approach:

> I've been looking at content management systems, more specifically open-source ones, and again find myself increasingly impressed with what I see. One of the beneficial side effects of open source development [...] is that the technical architectures tend to be modular which means that the core functionality [...] can be enhanced by additional functional modules. Add a vibrant user and support community and you get an explosion of *optional* additional functionality. Note the emphasis on *optional*. As a result, a barebones content management system will do just that. But want a weblog, instant messaging, discussion board, Wiki et al then, sir/madam, select from this list and voila! you've got something that begins to look awfully like a virtual learning environment. Which begins to propose a very interesting question,

> i.e. if that's the case, instead of a proprietary VLE why not use an [*sic*] such an extensible content management system over which you have some control? [Emphasis in original] [119]

In fact, Morrison then goes on and discusses Plone as an example of a suitable content management system.

Bergstedt et al. [16] note similarities between CMSs and e-learning systems and find e-learning systems to be lacking functionality commonly available in CMSs (see section 2.3). However, Bergstedt et al. come to a different conclusion in that they propose to transfer features from CMSs to e-learning platforms.

There is more research on architectures for e-learning platforms. The IEEE Learning Technology Standards Committee has issued an IEEE standard for *Learning Technology Systems Architecture (LTSA)* [68]. The standard describes itself as specifying "a high-level architecture for information technology-supported learning, education, and training systems that describes the high-level system design and the components of these systems." [68, p. 1] However, it does not describe a system architecture but rather defines abstract "system components," such as the "learner entity," and "processes," such as "coach" or "evaluate," in rather vague terms.

Avgeriou et al. [8] propose a "layered component-based architecture" based on the LTSA and describe the implementation of a prototype. While their proposal also uses a component-based architecture, it has few similarities with the eduComponents approach, as it does not abstract the functionality from the usage domain and reimplements numerous components which are *not* e-learning-specific, among others user management, a calendar, an e-mail client, chat client and server, and FTP client.

Numerous researchers have noted that modular architectures (such as component-based or service-oriented architectures) would, for obvious reasons, be advantageous for e-learning platforms. Especially with the increasing popularity of Web services and service-oriented architecture (SOA) [see, e.g., 131, 180] various proposals for service-oriented e-learning platforms have been made.

Liu et al. [101] observes that there are standards for information models, such as learning object metadata [69], learner profiles, and content packaging [70], and the LTSA as a standard for a conceptual component model, but that "there is a lack of an implementable architecture to define how to combine the information model with the component model" [101, p. 717]. The authors outline a "service architecture for building standard-driven distributed and interoperable learning systems," but do not give any details on the implementation.

Gehrke et al. [58] outline a very similar approach based on Web services.

In their article titled *Service-Oriented E-Learning Platforms: From Monolithic Systems to Flexible Services* Dagger et al. argue that "the traditional LMS is failing to keep pace with advances in Internet technologies and social interactions online." They describe three generations of e-learning platforms: First-generation monolithic systems, second-generation

modular systems (these are the current systems), and next-generation service-oriented systems. The authors make a bold prediction:

> The LMS vendor of old will no longer sell monolithic, one-size-fits-all solutions, but rather interoperable platforms and a range of e-learning services, letting consumers choose the right combination of services for their requirements. [37, p. 30]

However, this vision is not backed by a prototype or by more specific technical details that would show how it may be achieved.

Paulsson [133] suggests "a modular approach that includes both digital learning content and VLEs." Paulsson's work is motivated by an analogy with learning objects (small, reusable chunks of learning content[8]):

> [T]here is a need for Virtual Learning Environments (VLE) with similar characteristics as Learning Objects: a VLE that is modularised and composed out of small chunks of functionality that can be aggregated into a VLE that is adapted to a specific learning context, working in symbiosis with the chosen Learning Objects. [132, p. 1935]

Paulsson has implemented an experimental service-oriented framework called Virtual Workspace Environment or VWE. He mentions that "one of the most important design decisions is whether to implement a number of general services that learning services can implement, or whether to specify a larger number of e-learning specific services" [132, p. 1939]; Paulsson and Berglund advocates the first alternative for VWE and compares this to the approaches of the Open Knowledge Initiative (O.K.I.)[9] and the British-Australian E-Learning Framework (ELF)[10] projects; the former has a similar approach, i.e., providing more general services, whereas the latter focuses on pedagogical services that are defined in detail.

There are various research efforts aiming at creating models to bridge the differences between the architectures of existing e-learning platforms. Schewe et al. [152] describe a purely theoretical "conceptual view of Web-based e-learning systems." Liu et al. [101] suggest a functional model of e-learning systems based on a service-oriented architecture, emphasizing the "implementability" of the architecture, but apparently there is no actual implementation of their model. In a similar vein, Bizoňová and Ranc [20] propose a "metamodel" to enable the exchange of material between platforms despite different architectures.

3.3.2 Systems

Two production-quality e-learning platforms share some ideas with the eduComponents: OLAT[11] and Sakai[12].

8. See section 6.2 for a short discussion of the concept of learning objects.
9. `http://www.okiproject.org/` (accessed 2008-10-24)
10. The ELF project ended in 2007 and is succeeded by the e-Framework project `http://www.e-framework.org/` (accessed 2008-10-24)
11. `http://www.olat.org/` (accessed 2008-10-24)
12. `http://sakaiproject.org/` (accessed 2008-10-24)

OLAT, developed at the University of Zurich has already been discussed in some detail in section 2.5.1.

According to the project Web site, the Sakai project began in 2004 when Stanford University, the University of Michigan, Indiana University, MIT, and the University of California, Berkeley began building a common "courseware management system" rather than continuing their homegrown systems or licensing software from a commercial vendor. The Mellon Foundation provided initial funding for the project. At the time of this writing, Sakai is—again according to the Web site—in use at over 100 educational institutions, primarily in the United States, the Netherlands, and Great Britain.[13]

OLAT uses a component-based architecture i.e., OLAT is actually an application consisting of several components on top of a general-purpose framework. Sakai is said to be based on a service-oriented architecture [e.g. 162], there are, however, few details available. Davies and Davis [39] state that "[w]hilst Sakai is not currently using a service based paradigm, its designers are aware of the value of service architecture and recognize that Sakai may have service interfaces inserted in the future." Effectively, Sakai seems to be using a Java-based architecture similar to OLAT.

Both systems, OLAT and Sakai, recognize that there is basic functionality which is not domain-specific, and that the functionality specific to e-learning should be separated from it and built in a modular fashion on top of the basic infrastructure. This view is similar to that underlying the eduComponents; however, both systems have developed their own infrastructure and have reimplemented a large amount of functionality not specific to e-learning, such as wikis, blogs, chats, and other communication and collaboration functions.

3.3.3 Educational Uses of Plone

The eduComponents are based on the Plone content management system. Plone is well-established as a CMS in educational institutions. There are also some projects which—similar to the eduComponents—use Plone for uses more specific to education.

Rhaptos[14] is the software used for Connexions[15], an "environment for collaboratively developing, freely sharing, and rapidly publishing scholarly content on the Web," i.e., a public repository of teaching and learning materials, developed at Rice University and supported by the William and Flora Hewlett Foundation.

eduCommons[16] is a Plone-based system designed specifically to support OpenCourseWare projects. OpenCourseWare was initiated by MIT to publish MIT course content, such as lecture notes, exams, and videos on the Web.[17] This initiative was joined by other univer-

13. The Sakai Web site gives numbers varying from 100 to 200 institutions; see also `http://sakaiproject.org/portal/site/sakai-community/page/d89dabbf-a033-412f-80c4-a38931056b26` (accessed 2008-10-24).
14. `http://cnx.org/aboutus/technology/released` (accessed 2008-10-24)
15. `http://cnx.org/` (accessed 2008-10-24)
16. `http://plone.org/products/educommons` (accessed 2008-10-24)
17. `http://ocw.mit.edu/` (accessed 2008-10-24)

sities, organized in the OpenCourseWare Consortium[18]. MIT uses its own software for OCW, which is not available to other universities. eduCommons provides the functionality necessary to build and manage an open access collection. Utah State University, for example, uses eduCommons for their OpenCourseWare.[19] Development of eduCommons is funded by the William and Flora Hewlett Foundation and the U.S. National Science Foundation.

There have been some attempts to develop a Plone product for multiple-choice tests. Before starting the development of ECQuiz, we evaluated the Zope products Exam[20], XQuizz[21], and Survey[22]. Apart from missing functionality, while these products are based on Zope, they do not take advantage of the functionality offered by Plone.

There are two Plone products, which implement multiple-choice tests: eXam2go[23] and LTOnlineTest[24].

With the possible exception of eXam2go, all products are still in early stages of their development. Development of all products seems to have halted.

A now-defunct Austrian company was for some time offering pre-packaged Plone installations geared towards educational institutions under the name EduPlone [14, p. 217]. It seems that this company was also involved in a project[25], now seemingly abandoned, which aimed to develop products for Plone in educational settings.[26]

The *Plone4Universities* Web site[27] aims to collect Plone products for use in university environments.

3.3.4 Summary

While there is some theoretical research on architectural models—in particular proposals for service-oriented architectures—for e-learning platforms, there are not yet many implementations available. Most publications naturally emphasize the benefits of modularity, however, as Paulsson and Berglund [132] note, the focus is mainly on "exposing functionality from legacy systems as services, and rarely on new modular [...] architectures, where complex functionality can be constructed by aggregating simple, loosely coupled, services."

Some production-quality e-learning platforms, most notably OLAT and Sakai are based on component-based or service-oriented archi-

18. `http://www.ocwconsortium.org/` (accessed 2008-10-24)
19. `http://ocw.usu.edu/` (accessed 2008-10-24)
20. `http://zope.org/Members/J.A.R.Williams/exam` (accessed 2008-10-24)
21. `http://zope.org/Members/gillou/XQuizz` (accessed 2008-10-24)
22. `http://zope.org/Members/jwashin/Survey/` (accessed 2008-10-24)
23. `http://www.janus-projekte.de/exam/` (accessed 2008-10-24)
24. `http://lawtec.net/projects/ltonlinetest/` (accessed 2008-10-24)
25. `http://sourceforge.net/projects/eduplone/` (accessed 2008-10-24)
26. We can only speculate about the reasons for the foundering of EduPlone and other projects. A lack of developer resources is a frequent problem in open-source projects and is a likely cause; the projects apparently also failed to communicate the specific advantages of a solution based on Plone and may thus have been unable to attract enough interest.
27. `http://www.plone4universities.org` (accessed 2008-10-24)

tectures.[28] In principle, components can be added to or removed from these platforms; however, since they are explicitly targeted at e-learning, so that there is only a small "market" for components—this is in contrast to CMSs, which are used in a wide variety of applications. It also seems that most additional components are developed by universities to interface to campus management or legacy systems.

Plone is being used in educational environments, primarily as a content management system, not as an e-learning platform.

With regard to the main ideas realized in the eduComponents approach we can summarize the related work outlined above as follows:

1. Both currently available e-learning platforms and architectures proposed in research publications generally fail to abstract the functionality used in e-learning from its domain context. It thus seems to be generally accepted that an e-learning platform needs, for example, its own communication facilities (e-mail, forums, chat, etc.), even though their functionality does not differ from other implementations.

2. While proposed architectures envision for operators of e-learning platforms to be able to mix and match e-learning services from different sources according to their requirements (see the quote from Dagger et al. on page 73 above), this is not yet possible at the moment.

28. We find it debatable whether Sakai's architecture is actually service-oriented in a strict sense.

4 Assessment

In the preceding chapter we have briefly described the individual eduComponents products. We notice that, with the exception of ECLecture, all of them are concerned with assessment: ECQuiz, ECAssignmentBox, ECAutoAssessmentBox, and ECReviewBox implement various types of assessment functionality. This illustrates again that assessment functionality is the major feature which is required for e-learning, but which is lacking from general-purpose CMSs.

In this chapter, we will first describe *educational taxonomies*, which may serve as tools for classifying learning and assessment objectives. We will then give an overview of computer-supported assessment—or *e-assessment*—and its historical development. This forms the background for the subsequent detailed presentation of the assessment-related eduComponents products.

4.1 Bloom's Taxonomy

Bloom [21] defines a taxonomy of educational objectives, commonly known as *Bloom's Taxonomy*. Bloom's Taxonomy describes a number of *cognitive levels*; learning and corresponding assessment tasks can be classified according to these levels.

The six cognitive levels of Bloom's Taxonomy are:[1]

Knowledge Objectives on this level are about remembering and recalling information previously given to students. In assessments on this level, students are typically asked to recall, to define, to recognize, or to identify facts, terms, or basic concepts.

Comprehension Objectives on this level require an *understanding* of facts and ideas, i.e., it is not just about recalling facts. Students are typically asked to demonstrate comprehension by comparing, translating, restating, illustrating, interpreting, or explaining knowledge.

1. The following summary is based on [21], [94, p. 67], and [67, pp. 18–21].

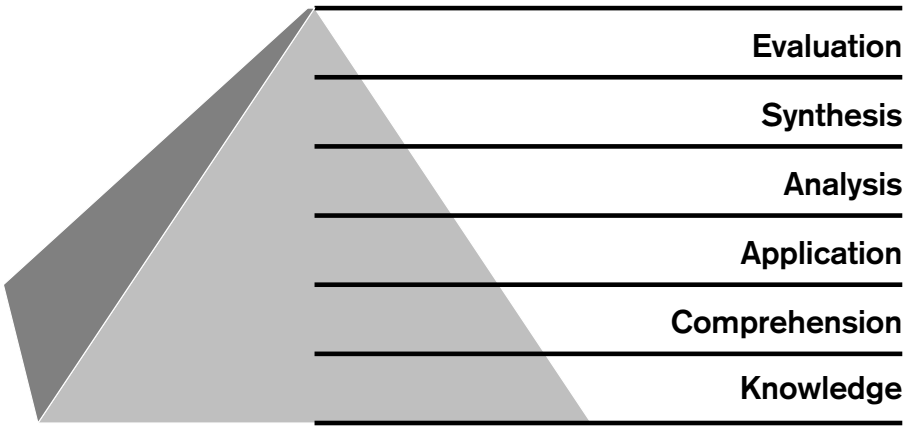


Figure 4.1: The levels of Bloom's Taxonomy are frequently depicted as forming a pyramid, emphasizing the assumption that each level must be mastered to reach the "top." Design adapted from [40, p. 128].

Application On this level, students are expected to solve problems in new (or unfamiliar) situations by applying acquired knowledge, facts, techniques, and rules in a different way.

Analysis Analysis refers to the ability to examine material and to break it down into its constituent parts. This requires students to identify parts, to analyze the relationship between parts, and to understand the organizational principles involved. For example, students should be able to distinguish between facts and inferences, to recognize unstated assumptions or non sequiturs, or to analyze the organization of a piece of work, e.g., in art or literature.

Synthesis Synthesis is the ability to combine parts into a new whole. Learning objectives on this level focus on creativity. Examples are: Writing a creative essay or a computer program, designing a research plan, or proposing alternative solutions for a problem.

Evaluation Evaluation is the ability to judge the value of material for some purpose or according to some set of criteria, and includes the ability to state why a specific judgment was reached. Students may be given a set of criteria, or the definition of criteria may be part of the task. Criteria may be internal, such as organization and consistency, or external, i.e., with reference to some set of standards. Evaluation tasks require students to appraise, to weigh, to judge, to decide, and to argue. Objectives on this level thus include aspects from all other levels, and add the ability of making judgements on the basis of clearly defined criteria.

Bloom and his collaborators considered the levels to form a hierarchy, going from least difficult—Knowledge—to the most difficult—Evaluation. The taxonomy also includes the assumption that students must master each level before they can proceed to the next one. This assumption is reflected in the typical visualization as a pyramid, such as the one shown in figure 4.1.

Bloom's Taxonomy has been widely adopted and is almost ubiquitous in education. It is not without criticism, though. For example, as Johnstone [83] points out, with respect to assessment tasks, the hierarchy does not necessarily reflect difficulty: There can be difficult knowledge questions and easy evaluation tasks. The exact level may also depend

Knowledge Dimension	Cognitive Process Dimension					
	Remember	Understand	Apply	Analyze	Evaluate	Create
Factual						
Conceptual						
Procedural						
Metacognitive						

Table 4.1: Anderson and Krathwohl's Revised Bloom's Taxonomy.

on the individual student: Depending on the student's knowledge, a task may be Application, Comprehension, or even Knowledge. Similarly, Davis et al. [40] admit that the taxonomy is "useful for examining the tasks we set for children," but warn that the model has "limited practical value when it comes to preparing lists of questions for actual learners" [40, p. 129].

Also, as it was published in 1956, Bloom's Taxonomy is now over 50 years old. On the basis of newer research there are proposals for new or updated taxonomies of educational objectives. Section 4.2 will give a brief overview of two newer approaches.

4.2 Newer Taxonomies

As we have noted above, Bloom's Taxonomy reflects the state of the art in educational psychology of 1956. As research has progressed over the last 50 years, cognitive psychologists have proposed a variety of new or updated taxonomies.

The best-known alternative proposals are probably the Revised Bloom's Taxonomy by Anderson and Krathwohl [3] (2000) and Marzano and Kendall's New Taxonomy [109, 110] (2000/2006).

4.2.1 Revised Bloom's Taxonomy

Anderson—a former student of Bloom's—and Krathwohl—a collaborator on the original taxonomy—started from Bloom's original taxonomy with the goal of updating it and of correcting some of the problems with the original taxonomy.

Unlike the 1956 taxonomy, the revised taxonomy makes the epistemological distinction between *knowing that*, i.e., the knowledge of facts and information, and *knowing how*, i.e., the ability to do something. This distinction is reflected in the two "dimensions" of the taxonomy, the *Knowledge Dimension* ("knowing that") and the *Cognitive Process Dimension* ("knowing how"). The Cognitive Process Dimension corresponds (with some changes) to the six levels of the original taxonomy, while the Knowledge Dimension is divided into four categories. Taking both dimensions together results in a two-dimensional grid, as shown in table 4.1, in which learning objectives can be localized.

The categories of the Knowledge Dimension are as follows:

Factual knowledge is knowledge of isolated pieces of information, such as names, dates, or terminology in some subject area.

Conceptual knowledge is knowledge of general principles, theories, models, or structures in a field. Thus, it is based on connections between the pieces of factual knowledge.

Procedural knowledge is knowledge of techniques, methods, procedures, or algorithms, and the knowledge about when and how to apply them to problems.

Metacognitive knowledge is knowledge about thought and learning processes, i.e., "thinking about thinking."

Compared two the 1956 version, there are also some changes in the Cognitive Process Dimension. The names of the levels were changed into verbs; see table 4.2. The lowest level was renamed from Knowledge to Remembering. The two top levels Evaluation and Synthesis were effectively swapped, and are now labeled Evaluating and Creating.

The new terms are defined as follows [from 3, pp. 67–68]:

Remembering Retrieving, recognizing, and recalling relevant knowledge from long-term memory.

Understanding Constructing meaning from oral, written, and graphic messages through interpreting, exemplifying, classifying, summarizing, inferring, comparing, and explaining.

Applying Carrying out or using a procedure through executing, or implementing.

Analyzing Breaking material into constituent parts, determining how the parts relate to one another and to an overall structure or purpose through differentiating, organizing, and attributing.

Evaluating Making judgments based on criteria and standards through checking and critiquing.

Creating Putting elements together to form a coherent or functional whole; reorganizing elements into a new pattern or structure through generating, planning, or producing.

On the basis of these definitions, the cells of table 4.1 on the previous page can be filled with concrete tasks, as shown in table 4.3 on the facing page. For example, a "*conclude* task" may require students to draw conclusions (*evaluate*) based on their knowledge of how a system works (*procedural knowledge*).

Table 4.2: Names of the levels in the 1956 taxonomy and in Anderson and Krathwohl's revised version.

Original Bloom's Taxonomy (Bloom [21], 1956)	**Revised Bloom's Taxonomy** (Anderson and Krathwohl [3], 2000)
Evaluation	Creating
Synthesis	Evaluating
Analysis	Analyzing
Application	Applying
Comprehension	Understanding
Knowledge	Remembering

Knowledge Dimension	Cognitive Process Dimension					
	Remember	**Understand**	**Apply**	**Analyze**	**Evaluate**	**Create**
Factual	List	Summarize	Classify	Order	Rank	Combine
Conceptual	Describe	Interpret	Experiment	Explain	Assess	Plan
Procedural	Tabulate	Predict	Calculate	Differentiate	Conclude	Compose
Metacognitive	Appropriate Use	Execute	Construct	Achieve	Action	Actualize

*Table 4.3: Anderson and Krathwohl's two-dimensional taxonomy annotated with "action verbs" suggesting typical student activities.**

4.2.2 Marzano's New Taxonomy

Another proposal for a new taxonomy of educational objectives was made by Marzano [108]. Like Anderson and Krathwohl's Revised Bloom's Taxonomy, his proposal aims to mitigate the shortcomings of Bloom's Taxonomy and to incorporate a wider range of factors that affect how students think.

Marzano's *New Taxonomy* is made up of three *systems of thought* and the *knowledge domain*. The knowledge domain comprises *information* (vocabulary terms, facts, time sequences, principles, and generalizations), *mental procedures* (procedural knowledge, such as knowing how to read a histogram or how to write a research paper), and *psychomotor procedures* (physical procedures, e.g., knowing how to dance or how to drive a car). The three systems of thought are (1) the *self-system*, (2) the *metacognitive system*, and (3) the *cognitive system*.

The self-system is thought to determine whether an individual will engage in (or disengage from) some task and to control the degree of involvement, or, in other words: It represents the motivation to learn something. Once the decision is made to engage in a task, the other systems of thought and the knowledge domain are activated. Four types of self-system thinking are relevant to the New Taxonomy [see 111, p. 22]: (1) examining importance, (2) examining efficacy, (3) examining emotional response, and (4) examining overall motivation. This basically means that the overall motivation to learn something depends on the perceived importance of the knowledge or skill, the perceived efficacy of acquiring it (i.e., whether one believes that one will be able to acquire it), and the attitude towards the knowledge or skill.

The metacognitive system is thought to be responsible for "monitoring, evaluating, and regulating the functioning of all other types of thought" [111, p. 21]. In the context of the New Taxonomy, it is assigned the following four functions: (1) specifying goals, (2) process monitoring, (3) monitoring clarity, and (4) monitoring accuracy, i.e., it is responsible for setting learning goals and monitoring the learning process.

The cognitive system, finally, covers the activities of knowledge retrieval (recognizing, recalling, and, in the case of mental and psychomotor procedures, executing knowledge), comprehension (identifying critical or defining attributes of knowledge), analysis (generating new information on the basis of knowledge), and utilization (applying knowledge for decision making, problem solving, experimentation, and investigation). [111, pp. 16–21]

* Based on the taxonomy table by Dianna Fisher, Oregon State University `http://oregonstate.edu/instruct/coursedev/models/id/taxonomy/` (accessed 2008-11-04)

		Levels of Processing					
		Cognitive System				Metacognitive System	Self-System
		Retrieval	Comprehension	Analysis	Knowledge Util.		
Domains of Knowledge	**Information**						
	Mental Procedures						
	Psychomotor Procedures						

Table 4.4: Marzano's New Taxonomy.

Like the Revised Bloom's Taxonomy, the New Taxonomy can be represented as a table (see table 4.4) where the three domains of knowledge (information, mental procedures, and psychomotor procedures) constitute one dimension and the levels of mental processing (the three systems of thought and, in the case of the cognitive system, the four subcomponents of that system) the other.[2]

One of the main differences between Bloom's Taxonomy and the New Taxonomy can be seen in the role of knowledge. In Bloom's Taxonomy knowledge is the lowest level of the educational objectives and basically consists in "little more than the remembering of the idea or phenomenon in a form very close to that in which it was originally encountered." [21, pp. 28–29]

Marzano and Kendall [111], however, note that Bloom included various forms of knowledge *and* the ability to recall that knowledge, such as specific facts, conventions, trends or sequences, classifications and categories, or principles and generalizations. Marzano and Kendall criticize the "mixing of types of knowledge with the various mental operations that act on knowledge" as "one of the major weaknesses of Bloom's Taxonomy" on the grounds that it "confuses the object of an action with the action itself" [111, pp. 2–3]. The New Taxonomy therefore separates the various types of knowledge from the mental processes (the systems of thought) that operate on them.

Thus, the cells of table 4.4 represent different uses of knowledge: For example, knowledge may be recalled or, in the case of physical procedures, executed (retrieval), or knowledge may be utilized as the basis of decision making in the metacognitive system.

4.2.3 Summary

In this section we have described two newer alternatives to Bloom's Taxonomy, Anderson and Krathwohl's Revised Bloom's Taxonomy and Marzano's New Taxonomy. A central aspect of both proposals is the different treatment of knowledge: Instead of treating it as the lowest cognitive level it is considered to be a separate "dimension" and cognitive processes (such as remembering, evaluating, etc.) are thought to *operate* on knowledge.

2. We have designed this table to allow for easy comparison with the Revised Bloom's Taxonomy (table 4.1); in [111, p. 2] the New Taxonomy is visualized as a cube, but since it actually has only two dimensions, we consider that visualization as misleading.

In contrast to Bloom's Taxonomy, which is now over half a century old, both the Revised Bloom's Taxonomy and the New Taxonomy can be considered to represent the current state of the art in educational research.

However, despite its age and its shortcomings, Bloom's original taxonomy can still be a useful tool if one is aware of its limitations. Its main value is in providing an established reference framework for describing and classifying the scope of learning objectives and assessments.

Since this dissertation is primarily concerned with the technical infrastructure for e-learning, we will use the terms from Bloom's Taxonomy in the following sections as it is more widely known outside of educational research than the newer proposals; however, we reinterpret its levels as a simplification of the processes in the sense of Anderson and Krathwohl's cognitive processes or Marzano and Kendall's levels of processing of the cognitive system.

4.3 E-Assessment

Assessment has always played an important role in education. Most, if not all, types of formal education use some sort of assessment, typically including a final exam to earn a grade, a degree, a license, or some other form of qualification.

Today, however, assessment is no longer restricted to grading at the end of a course (*summative assessment*), but it has been recognized that assessment is also useful for continuous monitoring of the learning progress (*formative assessment*), without being necessarily used for grading purposes.

Formative assessment, including *self assessment*, can play a vital role in motivating students since it provides them with a way to judge their own competency level and allows them to track their progress. It also enables students to identify areas where more work is required, and to thereby remain motivated to improve further. Of course, this requires that students receive feedback as quickly as possible.

Formative assessment also provides timely feedback for instructors, both with respect to the effectiveness of the course and the performance of the students; it thus helps to identify points which might need clarification.

For both groups, instructors and students, frequent testing is preferable. Case and Swanson [27] argue that infrequent testing makes each exam a "major event," with students investing much effort into preparation—they may even stop attending class to prepare for the exam. They also note that, with infrequent tests, students may be unable to determine whether they are studying the right material and with sufficient depth. Case and Swanson therefore conclude:

> Though it may be more time consuming for faculty, frequent testing reduces the importance of each individual exam and helps students to better gauge their progress. [27, p. 116]

In fact, assessment is always a time-consuming activity for instructors, especially if large numbers of students are to be assessed or, as in this case, if assessment is frequent. This has motivated the development of technical devices to support assessment, starting with relatively simple mechanical devices and evolving to today's *Computer-aided assessment (CAA)* or *e-assessment*.

Besides relieving instructors, e-assessment in particular offers new possibilities and new opportunities. For example, frequent formative tests are only practicable using e-assessment, and e-assessment also makes new or unorthodox assessment methods feasible, such as peer assessment[3].

However, automated assessment started with a more basic, now ubiquitous, assessment methodology, namely selected-response tests. While there are many possible types of selected-response tests, the best-known type is probably the classic *multiple-choice test*, which are particularly well-suited for automated assessment.

The following sections will give a brief historical outline of the development of automated assessment.

4.3.1 Early Automatic Testing Systems

The beginnings of automated assessment can be traced back to the 1920s, specifically to the automatic scoring of paper-based multiple-choice tests.

In the mid-1920s, Sidney Pressey, an educational psychology professor at Ohio State University, developed "teaching machines." These mechanical devices (see figure 4.2 on the facing page) offered drill exercises, and multiple-choice questions. Unlike systems for scoring tests, Pressey's Automatic Teacher was interactive and provided students with immediate feedback. [136]

In 1938, IBM announced the first commercial system[4] using *electrographic* technology, better known under the IBM trademark mark sense. Mark sense allowed candidates to mark their answers on regular paper, albeit on a separate answer sheet and using a special pencil. The pencil marks could be read by sensing the electrical conductivity of the graphite pencil lead, and the tests could be scored automatically without having to enter them manually into a tabulating machine or computer. Later, *optical mark recognition* (OMR) was, and is still being used to recognize the marks.

However, while these systems allow for automatic scoring, greatly accelerating the process, they are not *interactive* and thus do not provide *immediate* feedback for candidates.

In the 1960s, two types of systems appeared which can be considered the direct ancestors of modern e-assessment systems: *Computer-assisted instruction systems* and *electronic response systems*. Both types of systems were motivated by the ideas of *programmed instruction* and *programmed learning*.

3. See section 4.4.4 for a discussion of peer assessment and the eduComponents implementation of peer assessment in the ECReviewBox component.

4. The IBM 850 Test Scoring Machine, see `http://ibm.com/ibm/history/exhibits/specialprod1/specialprod1_9.html` (accessed 2008-10-17).

Figure 4.2: Exterior of Pressey Testing Machine, patent dates 1928 and 1930. (Photo from Wikipedia.)

4.3.2 Audience Response Systems

An electronic response system (also known as *audience response systems* or *classroom response systems*) allows students—in an auditorium or classroom—to respond to multiple-choice questions posed by the instructor by using an electronic sending unit. Modern sending units, also known as *clickers*, are typically portable and wireless. The units of the 1960s and 1970s were usually hard-wired keypads mounted at student seats. Judson and Sawada [86] provide a review of the literature on electronic response systems from 1968 to 2002 and point out: "Yet the aim was the same then as it is now—to obtain instant feedback from students in large classes."

Due to their use during face-to-face lessons, the motivation for classroom response systems is somewhat different from that of regular tests: The primary goals are to focus student attention, to enhance student involvement, and to provide immediate feedback for the *instructor*. In fact, Judson and Sawada [86] report on a number of systems from the 1960s and 1970s that provided special buttons for students to provide *student-initiated* feedback with respect to the lecture, e.g., "go faster" or "go slower." For example, Kumar and Rogers [96] emphasize this aspect as a distinguishing feature of their Olin Experimental Classroom (even though Judson and Sawada list a number of earlier references):

> Perhaps the most distinctive operational feature of the Olin Classroom is that students can respond at any time, on *their* initiative.

> A student who does not understand or is confused by what has just been presented or discussed can indicate his confusion as soon as he realizes that he is missing something. The traditional student response system permits the student to respond only at the instructor's initiative (that is, only when the instructor interrupts his presentation to pose a multiple-choice question) and then only to the particular question so posed (which may or may not be related to the students' confusion). [96, p. 189]

The authors go on to describe experiments for finding out the meanings to the keys of the "student response stations"; a preliminary list is given and includes items such as "clarify the objective or purpose" or "got it! proceed" [96, p. 190].

Of course, students also receive feedback on their answers to multiple-choice questions—either by the instructor or by indicators on the response console—, so that classroom response systems do provide some of the functions of tests, such as identifying gaps in knowledge. Even though the feedback for the individual student is perhaps somewhat indirect—as it is likely to be subordinated to the lecture process—the early electronic response systems nevertheless provided *immediate feedback* to students.

4.3.3 Computer-Aided Instruction

Computer-aided instruction (CAI) systems can be considered the direct ancestors of today's e-assessment systems. The first CAI systems were developed at the beginning of the 1960s. As mentioned above, the motivation for using computers in education came from the theory of *programmed instruction* and was directly based on Skinner's teaching machine, which was in turn inspired by Pressey's teaching machines (see section 4.3.1), or, as a contemporary paper summarized it:

> In a provocative paper published in 1954, B.F. Skinner reintroduced the concept of teaching machines. S.L. Pressey first introduced the idea in 1926, but because the *Zeitgeist* was not ready or because his writings lacked the persuasiveness of Skinner's, Pressey's ideas did not result in any extensive use of the new instructional method. [165, p. 481]

In the 1960s, however, the *zeitgeist* apparently *was* ready, and behaviorist theories, notably programmed instruction, became the dominant (theoretical) approach to teaching and learning. Programmed instruction heavily relies on self-teaching using a textbook or a teaching machine, with learning material structured into small units and presented sequentially. Students are allowed to proceed at their own pace; after each unit, students are asked questions they have to answer before they can advance to the next unit. Students receive immediate feedback to the correctness of their response.

This approach is obviously very well suited for computer implementation. The following quotation from a 1965 article illustrates this background of computer-assisted instruction:

> Now that programed instruction has emerged from the laboratories of experimental psychologists and become a bona fide

> teaching aid, thought is being given to the expansion and utilization of the media by which programs are presented. Probably the most far-reaching and spectacular of these ideas is the use of digital computers as teaching machines. It has been repeatedly stated that it is the program inside the teaching machine and not the machine itself that actually does the teaching. Research seems to indicate that students perform just as well using programed texts as they do using conventional teaching machines (Goldstein and Gotkin, 1962). Nevertheless, the versatility of computers opens a virtually unlimited area of research on learning as well as the potential of programing for individual differences. [45, p. 41]

Mainframe-Based Systems During the 1960s a number of CAI systems were developed; Dick [45], p. 42 stated in 1965 that "[a]t least five groups are now carrying out research on computer-based instruction." Among these systems, the PLATO system—already mentioned in section 2.2.2—stands out, as it evolved from the beginnings in PLATO I in 1960 up to PLATO IV in 1972. PLATO introduced a large number of innovations and PLATO IV can be considered a predecessor of modern e-learning platforms. In this chapter, however, we are only interested in the assessment features of the system.

The first version of PLATO, later referred to as PLATO I, was designed by Donald Bitzer in 1960 [18]. The system was described as a "teaching machine" and allowed a student (PLATO I only supported one student at a time) to control the sequence of materials (text and graphics) presented by the machine in the form of "slides." A course could also include slides on which one or more questions were posed to the student which had to be answered before they could proceed, see figure 4.3 on the next page.

Bitzer et al. describe the testing facilities of PLATO I as follows:

> The machine will accept and display constructed answers, as well as the more restrictive answers to multiple choice questions. The student is told as soon as he has submitted an answer whether it is correct or incorrect. In the latter case, the machine can indicate "NO" without in any way revealing the correct solution. [18, p. 160]

As the capabilities of the system went beyond simple true/false questions it also allowed inexact matching:

> Answers may be judged in a variety of ways. For questions which have a unique, well-defined answer, it is sufficient for the computer to compare the incoming student answer against the prestored correct answer. For questions whose answers require decimal expansions, tolerances can be specified within which the answer must lie. [18, p. 160]

The authors considered even more flexible "judging routines," which, however, would have required a more powerful computer.

Thus, PLATO I already included all basic components of a modern e-assessment system: A variety of question types, immediate feedback

Figure 4.3: "Screenshots" showing the assessment features of PLATO I (from [18]).

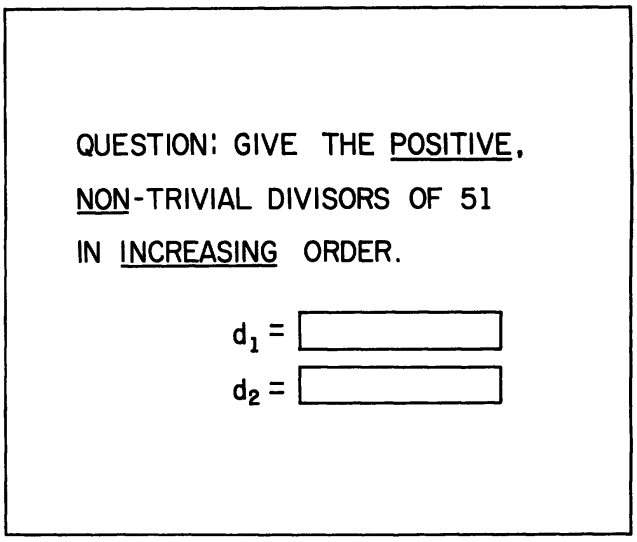


Fig. 4—Example of slide with questions.

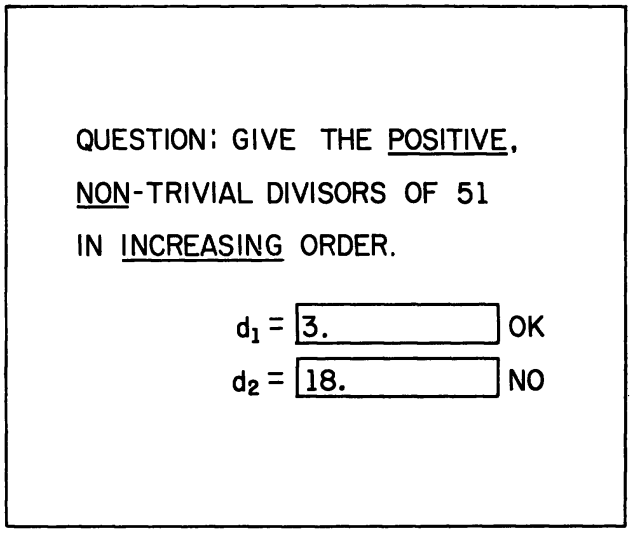


Fig. 5—Example of slide with student answers entered and judged.

for students, and instant reports of results in machine-processable form.

Even though the technology was available, giving candidates individual tests *and* immediate feedback was for a long time hampered by the price and the availability of terminals. For example, Palay [130], p. 101 notes in 1976 that "'[o]n-line' or interactive use requires numerous terminals and ports, and a computer system that is dependably 'up'," and goes on to calculate that giving 30 students an average of 20 minutes during a class period of 50 minutes would require "twelve (12) wholely [*sic*] dedicated terminals," concluding that "[e]ven with terminal costs declining, such an expenditure would be hard to justify." [130, p. 101]

Figure 4.4 on the facing page shows a screenshot of a multiple-choice test on the PLATO IV system. PLATO IV ran on Control Data main-

∗ *Diagnostic Quiz on Math Skills for Chemistry* (lesson `0mathskill`) by Ruth Chabay, University of Illinois, 1976. Screenshot taken by the author from the CYBIS system (see [185]) operated by Cyber1.org (see `cyber1.org` (accessed 2008-12-10)) using the Pterm terminal emulator.

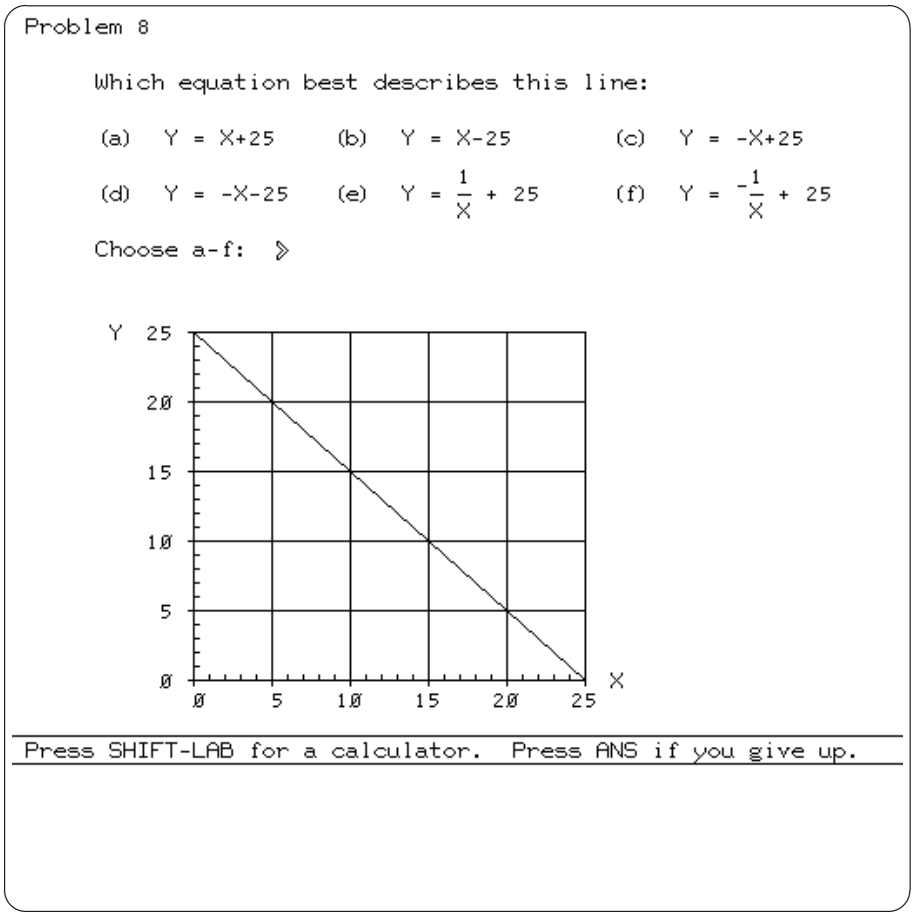


*Figure 4.4: Screenshot of a PLATO IV multiple-choice test.**

frames, which cost about 5 million U.S. dollars in 1972[5], according to Little [100], p. 77. While institutions using PLATO IV did not have to buy their own computers but used PLATO as a service, they still had to pay fees for connection and usage, and they required expensive terminals[6] This shows that, while the technology was available, it was hardly in widespread use.

Personal Computers The advent of microcomputers—later referred to as *personal computers*—in the 1980s, and especially the availability of cheap IBM PC clones in the mid-1980s, dramatically reduced the cost of computing.[7] This meant that more computers became available, and that computer access was no longer confined to dedicated terminal rooms. Thus, time and location restrictions for electronic testing were, to some extent, relaxed, and it started to become feasible and cost-effective to conduct tests for smaller groups without extensive advance planning and in decentralized locations, such as classrooms.

5. Roughly equivalent to U.S.$25 million at the time of this writing (see MeasuringWorth.Com), or about €16 million.

6. The exact cost is hard to determine. The Wikipedia entry on PLATO `http://en.wikipedia.org/wiki/PLATO` (accessed 2008-11-06) quotes a cost of "about U.S.$12,000"; contemporary sources, e.g., Thrush and Thrush [174] and Weiner and Emerson [183], cite sources which quote costs between U.S.$6000 and U.S.$10,500, while Small and Small [167] speak of U.S.$1000 per month.

7. A few important release dates for reference. Apple II: 1977; IBM PC: 1981; IBM PC XT: 1983; Compaq Portable (the first fully compatible IBM PC clone): 1982; Apple Macintosh: 1984.

For example, Thrush and Thrush in 1984 were enthusiastic about the new educational opportunities offered by microcomputers:

> If computer-assisted instruction is not a new concept, and if the use of large computers is only increasing slowly, then what accounts for the excitement? Technology has made yet another leap forward. With the advent of the microcomputer, almost the entire technological capability of the large computer is contained in a relatively small package at a price within reach of many schools and individuals. [...] For under U.S.$2,000 an effective educational system can be purchased that has unlimited mass storage, can draw and animate pictures, sing songs, monitor student progress, automatically average grades, print reports to parents, give and score tests, and even play 'Pac Man.'
>
> [174, p. 22]

With improved audio and video (or "multimedia") capabilities and increased processing storage capacities, PCs could not only replace and improve upon paper-based tests by offering immediate feedback and interactivity, but they now could also be used for self-paced tests involving recorded (i.e., non-synthesized) audio[8] or video prompts directly stored on the computer, which had previously required interfacing to analog videodisc players [see 103, p. 65].

Nevertheless, the availability of PCs represents more or less an evolutionary step for electronic testing. What revolutionized e-assessment was the *World Wide Web* and its associated technologies and standards.

4.3.4 Web-Based Assessment

The World Wide Web, or *Web* for short, was neither the first nor the most sophisticated hypertext system when it was announced in 1991. However, unlike Apple's HyperCard or EBT's DynaText products, it was non-proprietary and Internet-based.[9] And in contrast to Ted Nelson's Xanadu, it actually existed, one may add.[10] The possibility to deploy and develop server and client software independently and to add extensions without licensing restrictions was an important factor in the eventual success of the Web.

The *Gopher* system, developed at the University of Minnesota, was in many respects similar to the Web, released earlier than the Web and already in wider use. However, when the University of Minnesota announced in 1993 that it would charge licensing fees for the use of its implementation of the Gopher server, while the CERN (as the copyright holder) released the Web software into the public domain[11], interest rapidly shifted away from Gopher and towards the Web.

Tim Berners-Lee, the creator of the Web, asserts:

8. PLATO IV supported synthesized audio; Pembrook [134] reports on an application of the audio capabilities in music education.
9. see the comments of HyperCard developer Bill Atkinson in Kahney [87].
10. Theodor Holm Nelson coined the term *hypertext* in the 1960s and started Project Xanadu to implement his ideas. Theoretically, Xanadu is—in several respects—more sophisticated than Berners-Lee's World Wide Web. However, despite several attempts, nothing substantial has ever resulted from Project Xanadu.
11. `http://tenyears-www.web.cern.ch/tenyears-www/` (accessed 2008-10-17)

> CERN's decision to make the Web foundations and protocols available on a royalty free basis, and without additional impediments, was crucial to the Web's existence. Without this commitment, the enormous individual and corporate investment in Web technology simply would never have happened, and we wouldn't have the Web today.
> (`http://tenyears-www.web.cern.ch/` (accessed 2008-10-17))

The relationship of computer-based training (CBT) and educational authoring systems (mostly CD-ROM-based) to the Web was similar to that of commercial hypertext software to the Web: In the early 1990s numerous authoring systems for computer-based training (CBT) were available, with features more advanced than what was possible with HTML on the Web. However, like hypertext systems, CBT systems were highly proprietary and typically confined to standalone PCs or maybe local area networks. Locatis et al. [104] noted in 1992 that "[s]tandards for exchanging files and porting lessons to platforms using different computers and peripherals are being developed by vendors, trade associations, and standards organizations" [104, p. 78]—showing that interchangeability and portability were far from being a reality.

Albeit comparatively primitive, the Web offered a vendor- and platform-independent, non-proprietary, distributed hypertext system with basic facilities for interaction (forms and CGI scripts), and all the required software was also free—all this made the Web an appealing platform for educational applications, especially in academic environments. Instead of expensive proprietary authoring systems, typically only available on Microsoft Windows, with the courses also only being deployable on Windows PCs, authoring for the Web only required a text editor, or, as Locatis and Al-Nuaim understood it (perhaps not quite correctly) in 1999:

> Given recent advances in markup languages and their use for multimedia exchange on the Internet, any program capable of generating simple ASCII text can function as an authoring tool. [103, p. 63]

And not only could any system be used for authoring, the results were viewable on any platform for which a Web browser was available. NCSA Mosaic, the graphical browser that effectively triggered the explosive growth of the Web, became available in 1993, initially for UNIX (with sourcecode freely available for non-commercial use), with versions for the Commodore Amiga, Apple Macintosh, and Microsoft Windows following shortly afterwards.

It is interesting to note that the basic architecture of the Web is very similar to that of the mainframe-based systems, especially when considering HTML forms and CGI: The Web server takes the role of the host and the Web browser that of the terminal. With a block mode terminal (such as the IBM 3270), the terminal is sent a description of the form and the editable fields by the host. The terminal then renders and displays the form and permits the user to enter data into the form fields. Text entry is local to the terminal; only when the user has finished filling out the form, the terminal sends the entire user-entered data

Table 4.5: Evolution of technologies for multiple-choice tests.

Era	Technology	Effects
1920s	Pressey's teaching machines	Immediate scoring, but restricted
1930s	Mark sense	Accelerated scoring of paper-based tests
1950s	Optical mark recognition	Improved reliability
1960s	Mainframes, e.g., PLATO; audience response systems	Immediate feedback, but fixed time and place
1970s	Minicomputers, e.g., TICCIT	Cost reduction
1980s	PCs	Cheaper computers, thus more computers are available to instructors and students: Less place and time restrictions
1990s	Web	**Breakthrough:** Cheap computers and the Internet allow for immediate feedback almost anywhere, anytime—even more so as home Internet access becomes faster and cheaper
2000s	Mobile devices	Anywhere, anytime

(the field contents) back to the host in one transmission (a "block"). A Web browser displaying an HTML form works in exactly the same way.

It is hard to say when the Web was first used for quizzes, but it is not unlikely that they were one of the first applications of HTML forms[12]

4.3.5 Summary

We have tried to give a brief overview of the development of automated assessment in the preceding sections. The evolution of e-assessment is heavily influenced by the available technology: While multiple-choice tests have basically remained the same since the 1930s, technological developments have steadily removed limitations on when and where tests can be taken and feedback be provided to students. For example, with the mainframe-based systems of the 1960s, students could receive immediate feedback, but access to the systems was only available to few, and only during short time slots in special terminal rooms. Today's mobile devices provide Internet access—and thus also access to educational resources and tests—practically anywhere and at any time. Table 4.5 is an attempt to summarize this evolution.

Another development is that functionality that had been available on the pioneering systems of the 1960s, and which had been lost in the transition to PCs, is now available again, even though in different forms; Locatis and Al-Nuaim note:

> Computer-based education and the authoring tools for creating it have evolved from being networked—without multimedia, to being nonnetworked—with multimedia, to being networked again.

12. HTML forms were first standardized in 1995 with HTML 2.0 [17], however, Ragget had already defined most of the functionality in his 1993 HTML+ draft, which was in turn based on the implementation in NCSA Mosaic [see 140].

> While this has resulted in environments that function much like mainframe networks in the past, with added multimedia capability, the fundamental hardware and software are radically different. [103, p. 65]

4.4 Design and Implementation of the Assessment-Related eduComponents Products

As we have said above (see chapters 2 and 3), assessment is the central e-learning functionality missing from content management systems. Consequently, all eduComponents products (with the exception of ECLecture) implement various forms of assessment, allowing instructors to cover different educational objectives through a variety of assessment types.

We have already briefly presented the eduComponents products in section 3.1.2. In this section we will take a closer look at the assessment-related eduComponents products.

4.4.1 ECQuiz

ECQuiz allows instructors to create multiple-choice tests, or *quizzes*. Multiple-choice tests are especially useful as formative tests to quickly assess the performance of *all* students of a class without the need for extra grading work, or for self assessment.

There is a large number of systems providing Web-based quizzes. Most of these systems fall into one of two categories:

a) E-learning platforms with integrated assessment facilities. Most e-learning platforms (such as WebCT, Blackboard, Moodle, ILIAS, or OLAT) include quiz facilities.

b) Standalone assessment systems: The other category are standalone assessment systems, which concentrate on the delivery and evaluation of tests. Examples include Questionmark Perception[13], Hot Potatoes[14], Test Pilot[15], or TOIA[16].

The main advantage of quiz facilities built into an e-learning platform is obviously the integration with the courses managed in the same system.

Since the quiz facilities of e-learning platforms usually cannot be used on their own, i.e., outside of an online course[17], standalone quiz programs are interesting as a light-weight alternative for courses which are primarily face-to-face.

However, there are also instructors who consider the quiz facilities of e-learning platforms as too complex or otherwise unsuitable for their purposes; for example, Pouyioutas et al. [139] motivate the development of their own quiz software, even though their institution had

13. `http://questionmark.com/` (accessed 2008-10-17)
14. `http://web.uvic.ca/hrd/halfbaked/` (accessed 2008-10-17)
15. `http://clearlearning.com/` (accessed 2008-10-17)
16. `http://www.toia.ac.uk/` (accessed 2008-10-17)
17. *Online course* here refers to a unit managed by an e-learning platform.

acquired a WebCT license, with the need of faculty members for "simple-to-use software providing basic functionalities," after experience had shown that "[f]aculty members found WebCT quite complicated and therefore hard to use."

As the eduComponents in general, ECQuiz provides a solution that can be considered in-between the two categories listed above: On a "bare" Plone installation, ECQuiz can be used like a standalone quiz program. However, by using the CMS functionality of Plone (e.g., to manage and publish lecture notes or to provide discussion forums), other eduComponents products, or third-party products, ECQuiz can scale to a quiz facility integrated into a larger e-learning environment. Instructurs can thus start with quizzes and gradually use more and more e-learning functions but without being forced to adapt to the formal structure required by typical learning management systems.

The fact that ECQuiz makes quizzes available in a content management system has further advantages: Instructors who want to create tests need not learn how to use another system—if they know how to use the CMS to put together their Web pages, they can use the same system to create tests as "just another" resource. Students can then find the tests at the same place where the other Web resources are located, with the same look and feel.

In the following subsections we will describe the functions offered by ECQuiz in some more detail.

Creating Tests with ECQuiz

From the point of view of an instructor creating a quiz using ECQuiz, a quiz is a special type of folder containing test items, which are again folders holding a question and a number of answer choices. Furthermore, related questions can be grouped into *question groups*, which are then treated as a unit, both for editing and for delivery.

Thus, creating a test is technically the same as creating a collection of documents structured into folders in Plone, which means that instructors familiar with Plone immediately understand ECQuiz: Tests are just another content type, like texts, images, discussion boards, etc. Also, ECQuiz benefits from Plone features like timed publication, access control, metadata, full-text and metadata-based indexing and retrieval; like for other content types, discussion can be enabled for test items, which can be useful for collaborative test creation.

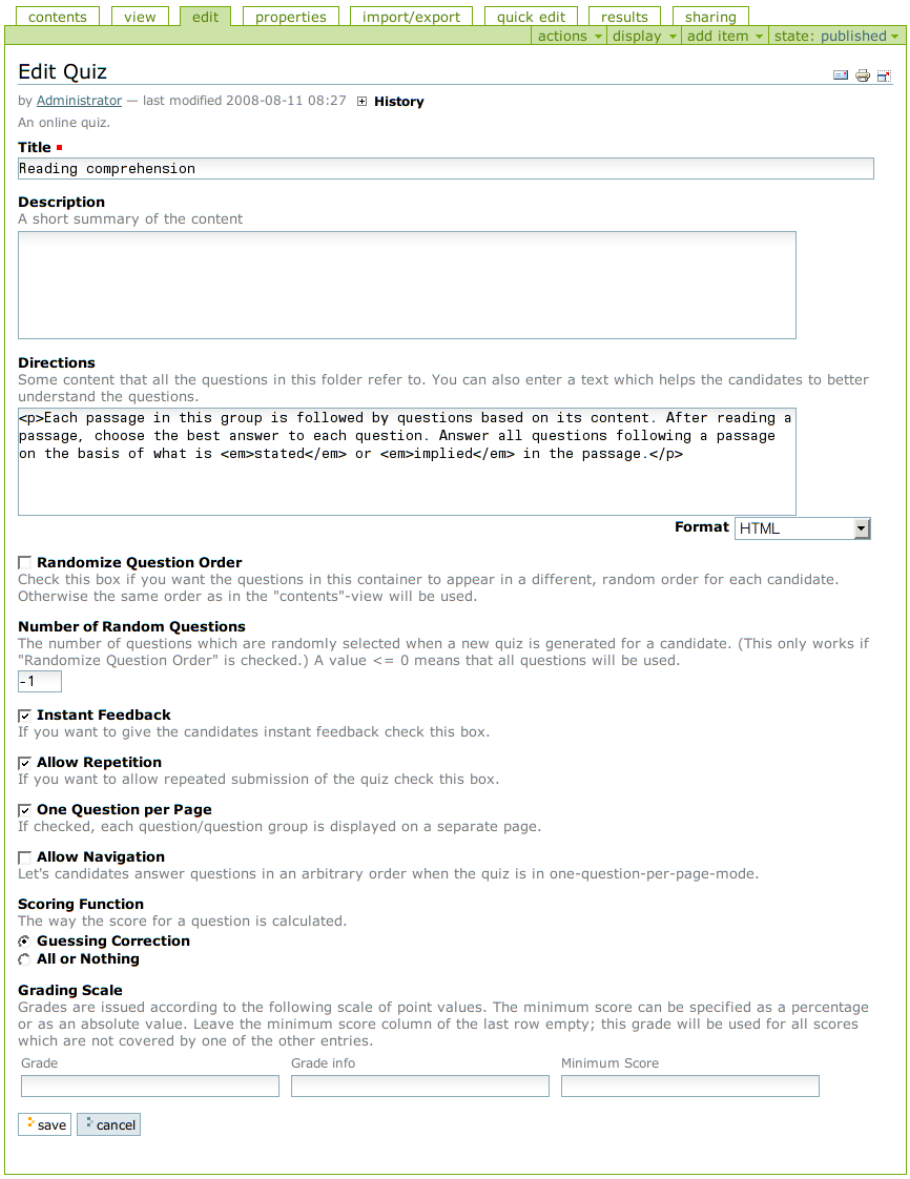
Instructors creating tests can set a number of options influencing the delivery of a test; figure 4.5 on the facing page shows the user interface for setting test options. These options are:

Randomize Question Order If this option is selected, the order of questions is random and different for each candidate.

Number of Random Questions If the number given here is greater than zero, the indicated number of questions is drawn randomly for each candidate from the questions contained in this test.[18] Otherwise all questions will be shown to candidates.

18. Thus, the test should have more questions than the specified number, otherwise all questions are displayed.

Figure 4.5: ECQuiz interface for editing metadata and setting options related to the delivery and scoring of the quiz.



Instant Feedback This option is intended for self-assessment quizzes. If it is selected, candidates get immediate feedback about their test score and their answer choices. Otherwise the results are only visible to the instructor.

Allow Repetition This option is also intended for self-assessment quizzes and enables candidates to repeatedly take a test.

One Question per Page This is a display option. By default, ECQuiz displays all questions on one page. When "One Question per Page" is selected, the test is spread over multiple pages, with each question (or question group) on a separate page. Figure 3.4 on page 68 shows the display of a test with this option enabled.

Allow Navigation This option influences the behavior of the "one question per page" mode; if it is deselected, candidates must work on the questions in the order in which they are presented. Otherwise, they can navigate among the pages of the test at will, and, for example, first work on page 5, then on page 1, etc.

The further options *Scoring Function* and *Grading Scale* influence the scoring of tests and the display of scores; these options will be described under the heading "Scoring and Results Analysis" below.

ECQuiz offers instructors a choice of question types; at the time of this writing the following types have been implemented:

Multiple-choice questions are the "classic" selected-response question type; both the single-answer and multiple-answer subtypes are possible.

Scale questions can be used to realize Likert scales and similar response types.[19] They can be displayed horizontally or vertically. Each answer is assigned a certain percentage of the total score for the question. For example, a 4-point Likert scale could have the following four choices with associated score percentages:

- Fully disagree, 0%
- Somewhat disagree, 33%
- Somewhat agree, 66%
- Fully agree, 100%.

Extended text questions allow free-text answers.[20] Unlike multiple-choice questions, this is not a selected-response type and answers cannot be scored automatically. In ECQuiz, extended text questions must thus be *tutor-graded*, i.e., assessed and scored by a human.

19. Likert scales have their origin in psychometrics, i.e., in personality tests; as such, they are not assessment items. However, similar question types, such as *best-answer questions*, where the alternatives differ in their degree of correctness, can be used for assessment.

20. The answers to extended text questions will usually be relatively short; the name comes from the fact that answers can consist of one or more sentences in contrast to *short answer* and cloze questions, where answers only consist of single words.

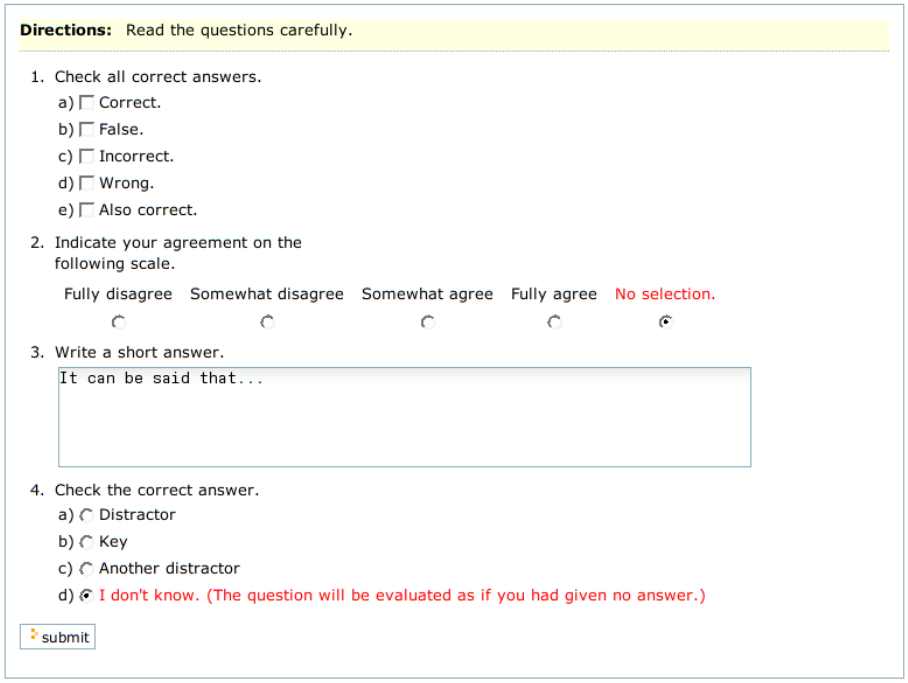


Figure 4.6: ECQuiz question types. This screenshot shows the appearance of the different question types available in ECQuiz; from top to bottom, these are single-answer and multiple-answer multiple-choice questions, scale questions, and extended text questions.

Figure 4.6 illustrates the appearance of the different question types. Figure 4.7 on the next page shows the editing view of a multiple-choice question with the options related to the question. In particular, the order of answers to a question can be randomized and answers can be randomly selected, similarly to the options for questions on the level of tests. All types of questions can be marked as tutor-graded, thus disabling automatic scoring of a question.

The smallest unit in ECQuiz are the answer choices. Figure 4.8 on the next page shows the view of an instructor editing an answer. Here, the text of the answer and its correctness are defined; answers can also be annotated with comments to be shown to candidates in instant-feedback mode.

Besides the question types described above, ECQuiz also offers a special reference type which can be used to include items from one test (or an item bank) into another test. Thus, for reusing items it is not required to copy them, but they can be referenced and will thus always reflect the current state of the master item.

As an alternative to creating tests through the Web interface of ECQuiz, tests can also be authored in textual form using the *Quick Edit* facility (see section 4.5.1). Individual questions and complete tests can also be imported and exported as files in IMS QTI (Question and Test Interoperability) format (see section 4.5.2).

Test Taking

The candidates' view of a test depends on the options selected by the instructor for this test, e.g., whether all questions are shown on a single page or whether the test consists of multiple pages, or whether navigation is allowed in multi-page quizzes.

As we have noted above, multiple-choice tests are especially useful as formative tests. A specific type of formative assessment is self as-

Figure 4.7: Editing view of an ECQuiz question.

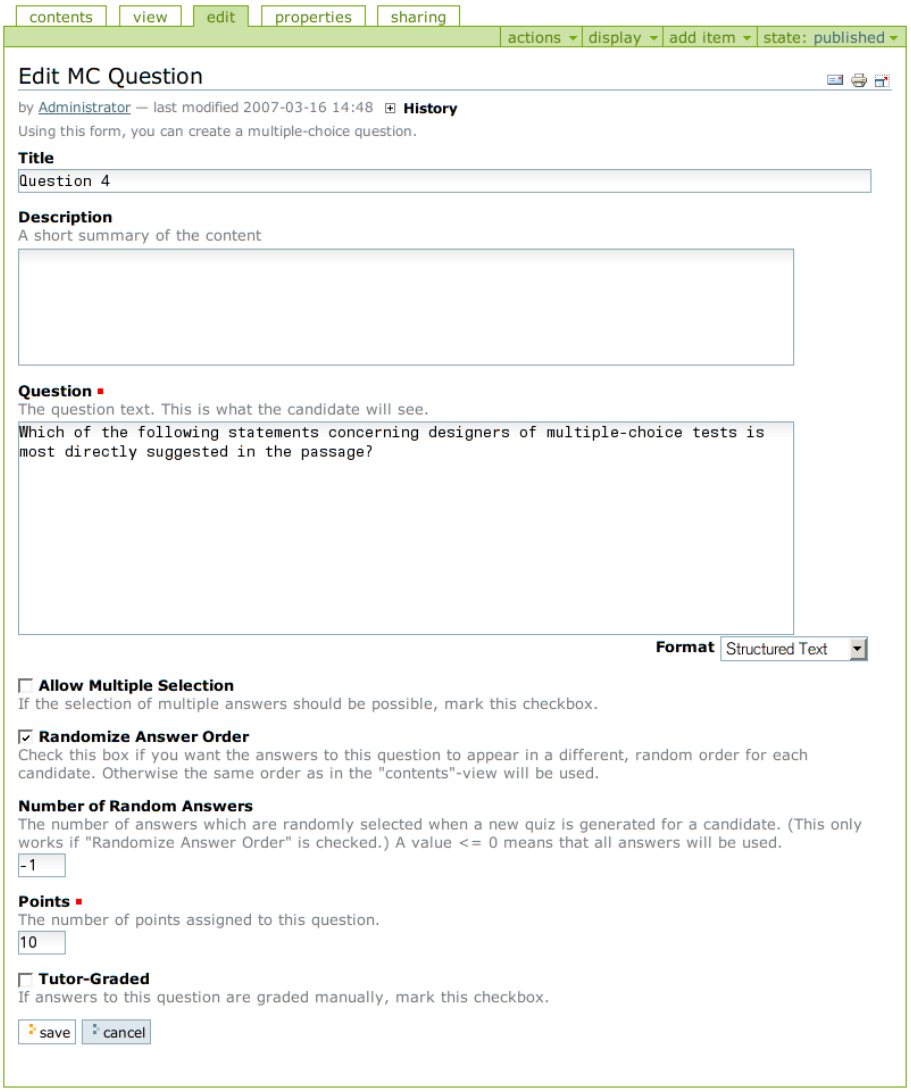


Figure 4.8: Editing view of an ECQuiz answer.

sessment, i.e., students evaluating their own performance and learning about their strenghts and weaknesses. For self-assessment tests, ECQuiz can provide students with instant results and question-specific feedback, and students can be allowed to take a test multiple times. Figure 4.9 shows an example of the feedback a candidate may receive for a self-assessment test.

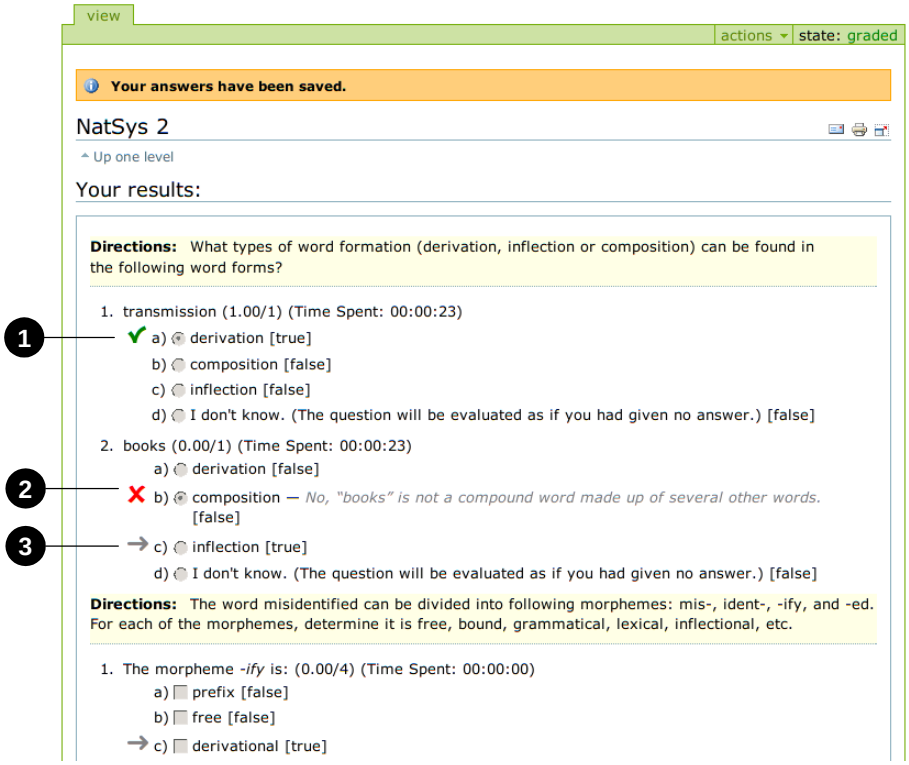


Figure 4.9: Example of the ECQuiz instant feedback option for self-assessment tests. ❶ is a correct answer, ❷ is an incorrect answer with additional feedback provided by the test author, and the arrow ❸ indicates the correct answer that the candidate should have selected.

Scoring and Results Analysis

The scoring of a test submitted by a candidate depends on several factors. The maximum score of a test is obviously determined by the maximum scores of the individual questions. The score a candidate receives then depends on their performance on the individual items and the method used to calculate the overall score.

Since a variety of scoring methods are conceivable, the first releases of ECQuiz[21] allowed instructors to define and upload their own *scoring functions* written in Python. However, since this facility constitutes a potential security risk, it was removed in release 1.0rc1 (February 16, 2006). We had further noticed that it was rarely used by instructors, if at all. Instead, the two most common scoring methods were built into ECQuiz, called *All or Nothing* and *Guessing Correction*.[22]

The scoring methods differ in the calculation of the score awarded for partially correct multiple-answer items. In the All-or-Nothing method, points are only awarded for a question if the candidate selected *all*

21. Then under the name of LlsMultipleChoice; the first publicly available version was released on June 14, 2005.
22. If necessary, other methods can easily be added, but for security reasons this cannot be done remotely.

and *only* the correct choices; in this case, all points will be awarded, otherwise, no points will be awarded.

When the Guessing Correction method is used, correct choices are awarded n/k points and each incorrect choice results in n/d points being subtracted, where n is the maximum number of points, k the number of keys (correct choices), and d the number of distractors (incorrect choices).

For example, if there's a maximum score of 10 and 2 correct choices and 3 incorrect choices, each correct answer is worth $10/2 = 5$ points and each incorrect answer subtracts $10/3 = 3.33$ points. Thus, when a candidate selects the two correct choices and a distractor, they would be awarded 6.67 points for that question.

If desired, instructors can specify a *grading scale* for the test (see figure 4.5 on page 95). A grading scale maps point scores to grades; the minimum scores can be specified as percentages or absolute numbers. In addition, informative text can be associated with each grade. Thus, besides for assigning "official" grades, this feature can also be used for adaptive feedback for different performance levels.

ECQuiz provides instructors with detailed results report, including the candidates' names, total score and item scores, grade (if assigned), and time needed for the test, providing an overview of the performance of all candidates. The report can also be exported for further processing using spreadsheet or statistics software.

Educational Objectives

Test scores can only indicate the degree to which a student has attained an educational objective with some accuracy if the form of assessment is suitable for the objective. While multiple-choice questions—and other forms of selected-response test items—have a number of practical advantages, they are obviously only appropriate when the achievement of the educational objective can be measured by having students select their responses from a set of given alternatives.

Multiple-choice questions are frequently used to measure lower-level objectives (in Bloom's Taxonomy), such as those based on the knowledge of facts, terms, and basic concepts, methods, and principles. However, multiple-choice questions can also be used for assessing higher-level objectives requiring comprehension, application, and analysis.

Thus, with correctly constructed test items, ECQuiz can be used for the assessment of the taxonomy levels Knowledge, Comprehension, Application, and Analysis. An example for a multiple-choice question assessing Application could be a question asking students to select the correct area of a geometric figure from a list of plausible choices would require them to apply a formula and calculate the area to be able to answer the question correctly.

The assessment of Synthesis and Evaluation is, however, difficult using selected-response items since these objectives on these levels require creativity. For assessing objectives for which selected-response items are unsuitable, the extended text question type of ECQuiz can be used; as long as the answer can be expressed textually, this question type

does not impose restrictions on the answer. Extended text questions cannot be scored automatically. However, ECQuiz allows instructors to combine extended text questions with selected-response items, so that for each objective a suitable question type can be used and the need for human scoring is limited to those items where extended text questions are required to assess the educational objective.

4.4.2 ECAssignmentBox

While ECQuiz allows for constructed-response items in the form of extended text questions, this functionality is primarily intended for assessment combining selected-response items with one or two extended text questions for covering educational objectives on the level of Synthesis or Evaluation.

ECAssignmentBox, on the other hand, is the eduComponents product specifically designed for the assessment of higher-level educational objectives. ECAssignmentBox implements the creation, submission, grading, and management of essay-like assignments. *Essay-like* means that the students' answers are more or less unrestricted, free-format text. If the file upload facility is used, students can submit any file, including non-text documents, such as audio, graphics, or specialized application files.

The assessment process is semi-automated, i.e., the assessment is done by the instructor, who is aided by the tool during the process of grading students' work and giving feedback. ECAssignmentBox supports instructors with management, statistics, and analysis functions for the submissions in assignment boxes.

For organizational purposes, assignment boxes can be grouped together using ECFolder to form "exercise sheets," courses, or similar units. ECFolder extends the statistics and analysis functions to all submissions in all contained assignment boxes.

Creating Assignments with ECAssignmentBox

The basic idea is that instructors create *assignment boxes*, which are a special type of folders, into which students submit their answers or solutions. The submissions are stored as *ECAssignment* objects inside the assignment box.

The assessment of student submissions can be described as a process that starts with the submission of a student's answer and ends with the assignment of a grade by the instructor. This process can be modeled as a *workflow*. ECAssignmentBox uses Plone's workflow facilities to define a specialized workflow for student submissions, i.e., from the initial submission to the final grading, submissions are put through a number of *workflow states*. The workflow is designed to accommodate different ways of handling submissions. It will be discussed in detail later, we will first describe the attributes of ECAssignmentBox objects.

When creating a new or modifying an existing assignment box, the edit view, as shown in figure 4.11 on page 103, is presented to the instructor. This form allows the user to set or modify all attributes of the ECAssignmentBox object.

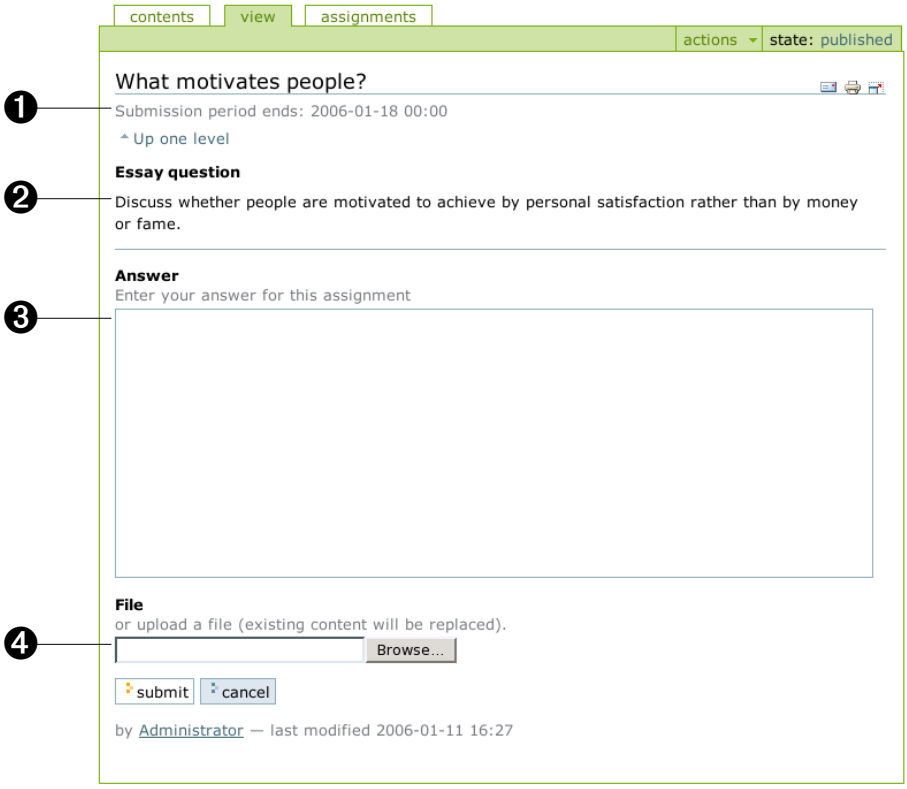


Figure 4.10: Student's view of an assignment box. At the top of the assignment box, students are informed about the submission period ❶. The main part consists of the the assignment text ❷and the answer field ❸; alternatively, a file can be uploaded ❹

Besides the standard Plone attributes (e.g., a title), ECAssignmentBox objects have a number of attributes, which, together, realize the specific assignment box functionality.

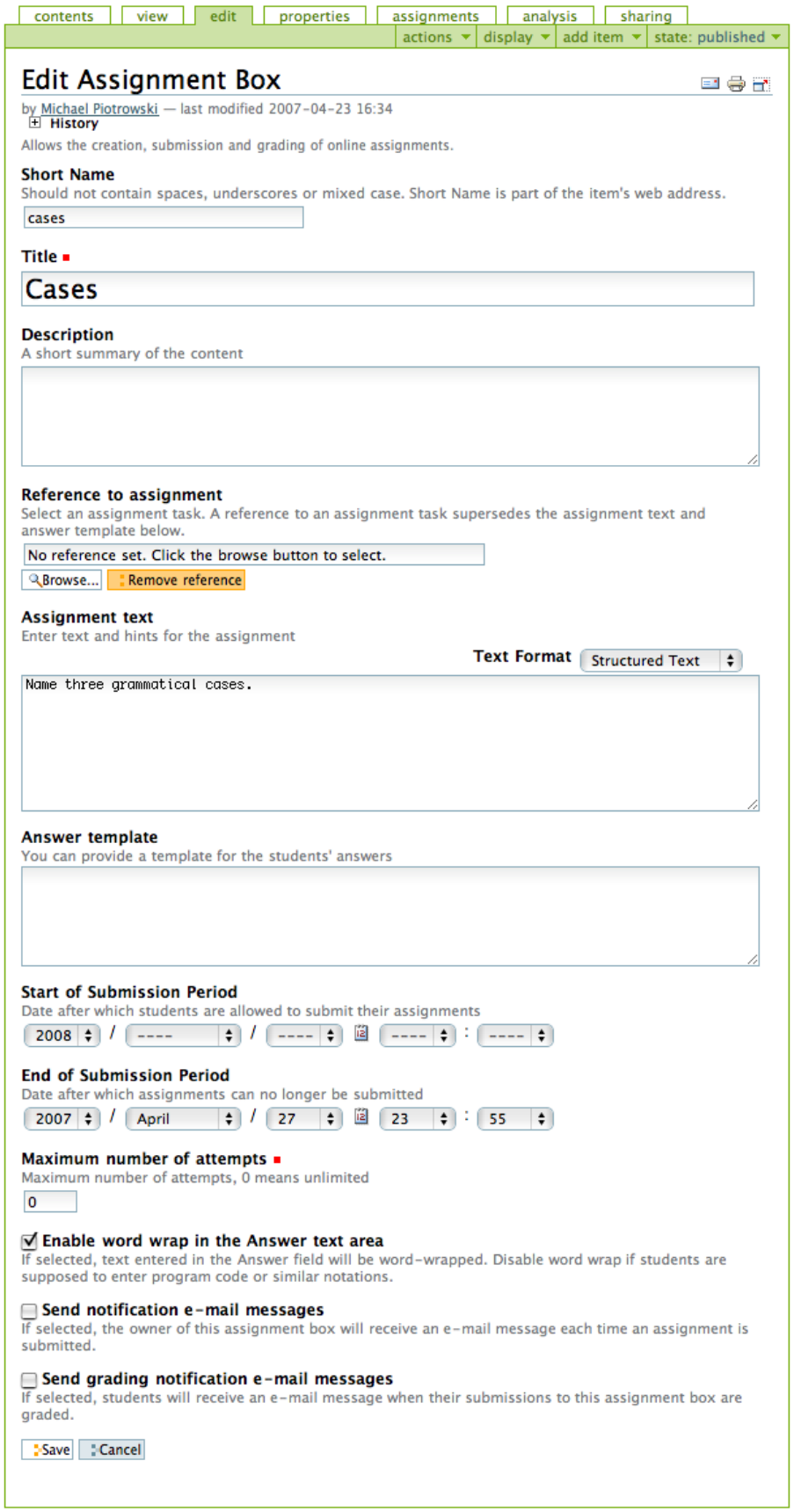
Assignment First, there is a field for the assignment proper, i.e., a description of a task the students are supposed to work on. The assignment is typically mostly textual, see figure 4.10. However, the assignment can not only consist of text, but it can contain anything possible in a Web page, so it can include pictures or graphics, but also audio or video clips. It is thus possible to create assignments referencing additional materials, e.g., a photo of a painting or a statue, an audio recording of a dialog, or a mathematical graph or a geometric figure.

Hence, ECAssignmentBox can not only be used to realize any essay-like assignment that can be realized on paper, but it can also be used to create assignments making use of multimedia content. In a traditional setting, this type of assignments could only be approximated in class (e.g., by showing a video to the class), but would be difficult to give as a homework assignment.

Answer template Next, the instructor can also specify a so-called *answer template*. The text specified in the answer template will appear as the default content of the answer field that is shown to students accessing the assignment box (see figure 4.10).

The answer template can be used to provide students with text they can or should use in their answer. For example, students can be guided

Figure 4.11: ECAssignmentBox editing view.



in their answer by providing an initial sentence fragment to complete, or a form-like structure to fill out (similar to cloze questions). In fact, the answer template can be used to realize most of the semi-open items (*halboffene Aufgaben*) described by Rütter [149], except that the "form" is not read-only [see also 15].

Since the answer field only allows for text answers, this field is currently also text-only. ECAssignmentBox offers the possibility of uploading rich-text or multimedia answers; if it is desired to provide an answer template that is not text-only, this could be realized by offering an editable file for download, into which students would insert their answers, and would then upload.

Submission period Instructors can specify a *submission period* for the assignment box, i.e., the time frame during which students are allowed to submit their answers. Before and after the submission period students can view the assignment (if it is published) and their submissions, but they cannot make new submissions.

Number of attempts The "Maximum number of attempts" attribute allows an instructor to control how often students may replace their submission during the submission period. For example, if set to "3," students may replace their initial submission two times. Superseding submissions is controlled through the ECAssignmentBox workflow (see below), in that previous submissions are preserved in the state 'superseded', but only the last submission can proceed through the workflow, i.e., is considered for reviewing and grading.

This feature is especially interesting for automatically checked submissions in ECAutoAssessmentBox (see section 4.4.3). For manually reviewed submissions, it can be used to reduce the number of superseded submissions, if desired for technical reasons. It may also be used to force students to submit *only* the version they consider final; there may be pedagogical reasons for doing this. On the other hand, the possibility of submitting draft versions during the submission period offers students a way to save intermediate states of their work if they write their answers in the Web browser, since otherwise the complete text would be lost if the browser crashes, or to pick up and continue the work on the answer later on from a different computer. Some of our students have indeed used this feature for this purpose, which we did not anticipate. Given that very much research and learning can be done online today, this seems to be very reasonable. For example, a student managing her references in CiteULike, has access to her personal library wherever she has Web access. Dictionaries and other reference works are available online as well. ECAssignmentBox currently supports this "ubiquitous mode" of working only incidentally; it would be interesting to further explore the possibilities.

Other options Depending on the expected type of answers, the instructor can enable word wrap in the answer text area, where students are entering their answers. If the assignment requires the student to produce, e.g., a narrative text, the instructor will leave word wrap en-

abled. On the other hand, if the answer will be in some formal notation or some other line-specific format, word wrap should be disabled.

ECAssignmentBox also offers two notification options: Instructors can choose to be notified when assignments are submitted. Instructors can use this to get an overview of student activity and of submissions before the end of the submission period. Instructors can also choose to have students automatically notified when their submissions have been graded.

The usefulness of the notification options depends on various properties of the course. If the course has a fixed time schedule, e.g., with weekly face-to-face sessions, notifications are less important, since both instructors and students will typically check the status at regular intervals to prepare themselves for the next session. On the other hand, in a pure online course, where participants may move ahead at their own pace, and with possibly no fixed submission periods or pre-scheduled appointments, it will be important to be notified of these asynchronous events.

Reusable Assignments: ECAssignmentTask

After several semesters of using ECAssignmentBox for all courses of the WDOK research group and after having created hundreds of assignment boxes, we had collected much experience. On one hand, the concept of closely associating the assignment, the assignment box, and the submissions had proven to be right: Submissions *only* exist in relation to some assignment, therefore it makes sense to associate them closely. On the other hand, with larger numbers of assignments and especially with large courses that are split into multiple groups of students, it showed some drawbacks.

For example, we may consider a course taught in three groups at different times (say, group 1 on Monday, group 2 on Wednesday, and group 3 on Thursday). The assignments in the three groups will be mostly identical, but the assignment boxes must, obviously, have different submission periods for the three groups. Also, students should only be allowed to submit their solutions in the group in which they are registered, so that everybody has the same time to work on their solutions. Thus, it is natural to set up three ECLecture objects for the three groups. Each ECLecture object then contains ECFolder objects and assignment boxes. Using the registration function of ECLecture and Plone permissions, it is no problem to control access.

However, each assignment box has to be copied for each group, which is tedious. What is more, when a change needs to be made to an assignment after copying (e.g., to correct an error or to change the wording), the change has to be executed for each copy of the assignment box. This process is both time-consuming and error-prone.

Another scenario is the reuse of assignments, which is only possible by copying an assignment box. If the assignment box is not an empty (i.e., without submissions) "master copy" explicitly created for copying into courses, it means that the assignment box *including* the submissions must be copied, and then the submissions must be deleted in the copy. Again, this is time-consuming and error-prone.

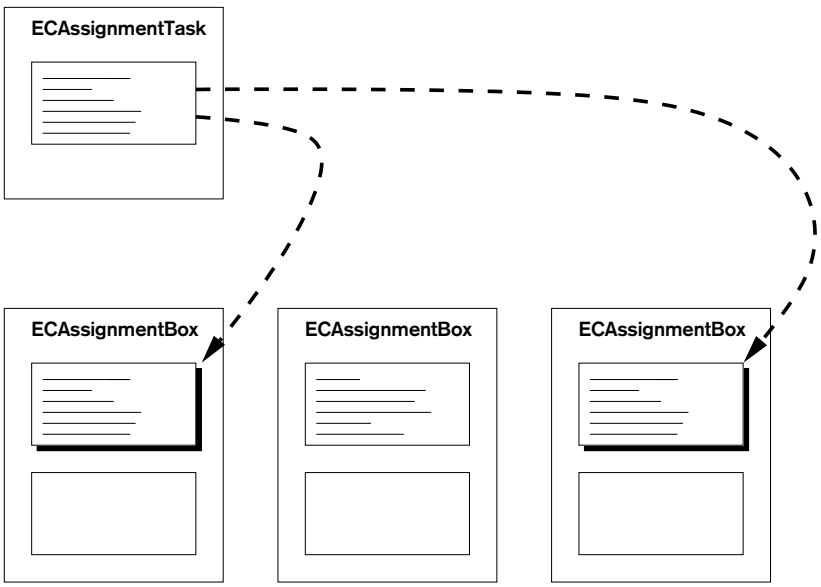


Figure 4.12: Relationship of ECAssignmentBox and ECAssignmentTask: Any number of assignment boxes can reference one assignment task.

With release 1.3 of ECAssignmentBox we therefore implemented an alternative to including the assignment in the assignment box: The assignment can instead be stored as a separate ECAssignmentTask object, which is then referenced by assignment boxes (see figure 4.12). When viewing the assignment box, the assignment defined in the ECAssignmentTask is "transcluded" into the assignment box and appears just like an assignment defined in an ECAssignmentBox object.

Putting the assignment into an external entity has many advantages, the most important being that it enables the creation of a central repository of reusable assignments.

The experience with ECAssignmentTask has yielded important insights about assignments, assignment boxes, and submissions. An *assignment* can be regarded as a class (in object-oriented terminology). In a concrete course, this class is instantiated in an *assignment box* as a concrete assignment, for which students have to submit their solutions. The concrete assignment differs from the abstract assignment in that it is embedded in the *context* of a course with certain goals, a certain target audience, etc. and preceding and following assignments, all contributing to the perception of the assignment.

It is obvious that the *same* assignment can be perceived as hard or easy by different target audiences. Thus, the difficulty of an item can be considered latent, manifesting itself only in the instantiation.[23] Consequently, when making this distinction, it becomes clear that some attributes of ECAssignmentBox described above really apply to the abstract assignment, while others apply to the instantiation in the assignment box. For example, the maximum number of attempts or the submission period are clearly an attribute of the assignment box, whereas the answer template can be considered part of the assignment.[24]

23. Of course, *difficulty* is a rather abstract term, referring to some average or intended difficulty; the actual difficulty depends on each student's knowledge and abilities.
24. Different answer templates are imaginable for one assignment text, however, as they are likely due to different pedagogical objectives, these variations should then be considered *different* assignments and treated as such.

On the implementation side, it is worth noting that ECAssignmentTask objects are folder-like, whereas one may expect them to be document-like. We decided to make ECAssignmentTask folder-like to enable instructors to store data related to the assignment but not intended for students to see, such as notes, references, or model solutions, but also student feedback with the assignment. Storing this type of information with the assignment keeps everything related to the assignment together and helps with reusing the assignment, since other instructors can quickly and easily find relevant background information. Currently only basic folder functions are available. However, this model lends itself to further extensions; for example, the performance of different classes for this assignment could be recorded.

The Students' View

Figure 4.10 on page 102 shows the students' view of an assignment box. Students can read the assignment text and enter their answer into the text field or upload a file. If the submission period is restricted, it will also be displayed when the submission period ends. ECAssignmentBox allows multiple attempts to answer assignments up to the maximum number of attempts specified by the instructor; in any case, submissions are only allowed until the submission period has ended, or until the submission has entered the Pending state.

The ECAssignmentBox Workflow

As we have mentioned above, the assessment of student submissions in ECAssignmentBox is semi-automated, i.e., the assessment is done by instructors, who are supported by ECAssignmentBox during the entire process of grading students' work and giving feedback.

ECAssignmentBox therefore defines a specialized workflow for student submissions (i.e., ECAssignment objects). The workflow is designed to accommodate different ways of handling submissions. The following workflow states are defined:

1. Possible states during the submission period; students are allowed to resubmit another version:
   - *Submitted*: An answer was submitted, but it may be superseded by a later submission. This is the initial state of an assignment.
   - *Superseded*: An assignment is automatically moved to this state if the student has submitted another assignment.
2. Possible states during the assessment process; students are no longer allowed to submit another version:
   - *Pending*: The assignment is under review.
   - *Accepted*: The assignment has been reviewed and has been accepted.
   - *Graded*: The solution has been reviewed, accepted and graded.
   - *Rejected*: The assignment has been reviewed and has been rejected.

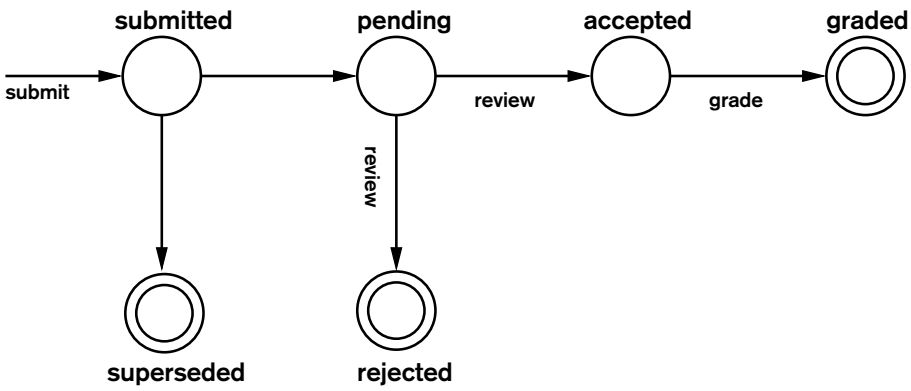


Figure 4.13: Diagram of the workflow states for ECAssignment objects.

Figure 4.13 is a diagrammatic representation of the workflow states.

Managing and Grading Submissions

ECAssignmentBox offers instructors a variety of functions for managing students' submissions. For instructors, assignment boxes provide two additional tabs, *assignments*[25] and *analysis*. The *assignments* tab allows instructor to see a list of all submissions to this box (see figure 4.14 on the facing page).

The *analysis* tab provides instructors with various statistical information for this assignment box, such as the number of submissions in certain states, the number of submissions per day, or the number of attempts students made (see figure 4.15 on the facing page).

Students only see the *assignments* tab, which they can use to view their own submissions, including the current workflow state and assigned grades. A speech balloon icon indicates that the instructor commented on the submission; by selecting the submission, the student can view the instructor's feedback.

On the level of an "exercise sheet" or a course, ECFolder and ECLecture[26] provide the information over all submissions (recursively) contained in the ECFolder or ECLecture object. In addition to the information given by ECAssignmentBox for an individual assignment box, ECFolder and ECLecture provide a *statistics* tab; the associated statistics page (see figure 4.16 on page 110) summarizes workflow states and grades for each student's submissions and allows to monitor the progress of a student based on planned[27] and completed assignments. Also, overall average and median grades are calculated.

Handling plagiarism

Plagiarism is increasingly considered a problem in education, especially due to the ease with which information can be found on the Internet or exchanged among students, copied, and assimilated into

25. The tab should be labeled *submissions*, but this change has not yet been made to avoid confusing existing users.
26. ECLecture objects provide the functions of ECFolder objects if the ECFolder product is installed.
27. Instructors can specify the number of planned assignments for a course and the state or states which indicate that a submission is completed.

Figure 4.14: Example assignment box evaluation view, showing students' submissions to that assignment and their workflow states and grades.

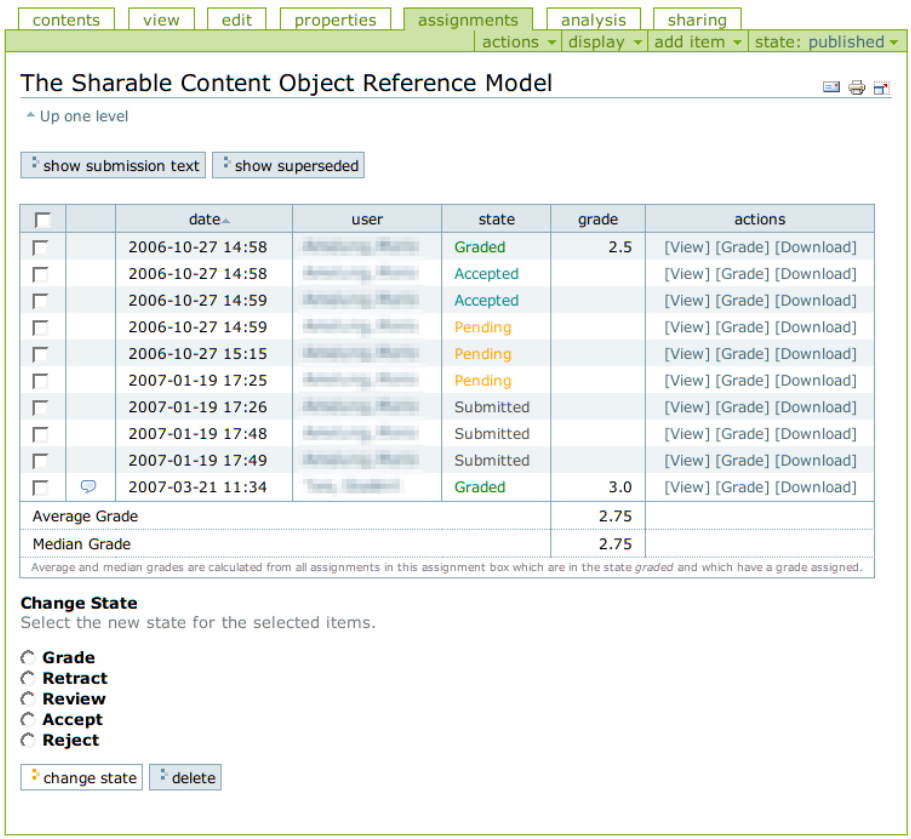
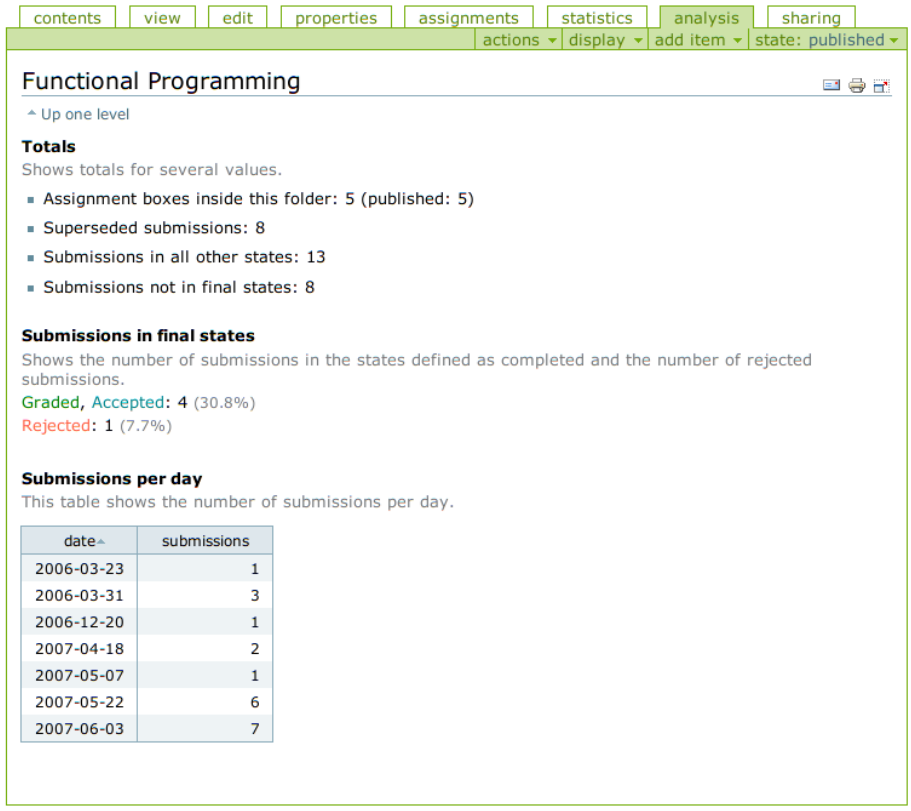


Figure 4.15: The analysis view provides instructors with various statistical information on the submissions made to an assignment box.



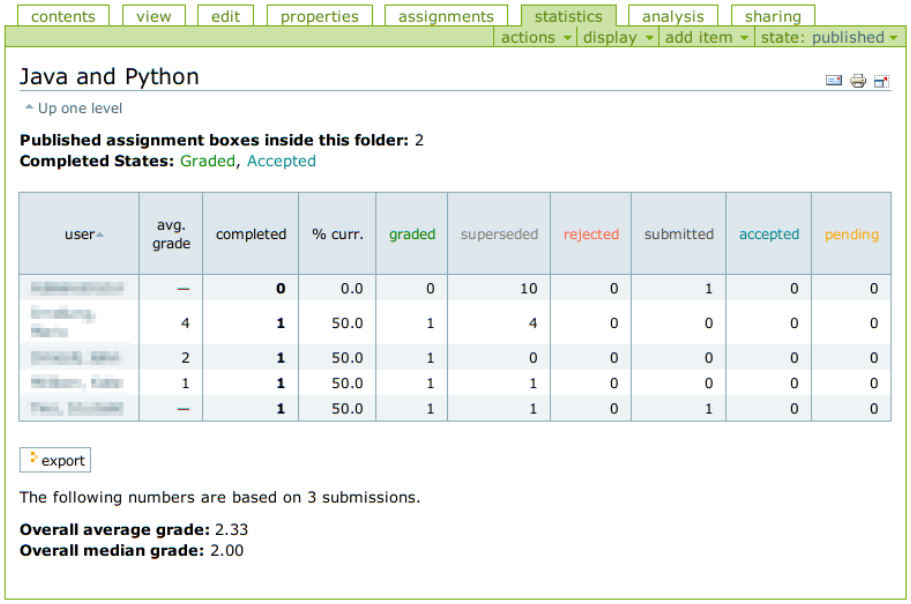


Figure 4.16: The statistics view of ECFolder presents high-level statistics about the submissions to all assignment boxes contained in the folder. Instructors can see information on all submissions, students can view information on their own submission.

submissions. Numerous approaches to automatically detect plagiarism can be found in the literature [e.g., 38, 126, 105, 81].

Plagiarism detection is not a topic of this dissertation. However, some research and development was done in relation to the handling of plagiarism in the eduComponents, since this is clearly an issue in education.

To help instructors with the comparison of submission, whether they were made by different students or whether they are revisions of one student's submission, we have implemented an experimental *diff* function, which enables instructors to select and compare two submissions in an assignment box. Figure 4.17 on the facing page shows an example screenshot of the prototype implementation.

The comparison function will typically be used when there is some indication that plagiarism may be involved. In a diploma thesis supervised by the author, Dervaric [42] developed a set of extensions to ECAssignmentBox for finding potential instances of plagiarism. These extensions, called PlagDetector, offer a variety of algorithms, tools, and visualizations which can help detecting plagiarism in ECAssignmentBox submissions.

Figure 4.18 on the facing page is a screenshot illustrating one of the available visualization options for similarities between a number of submissions. The screenshot shows a *categorical patterngram* (CP) at the top and a *composite categorical patterngram* (CCP) below. The CP shows the degree of similarity between one submission and the rest of the submissions for this assignment. The CCP shows which particular submissions are similar. See [146] for more details on CPs and CCPs.

Educational Objectives

ECAssignmentBox is a tool for assessing educational objectives through constructed-response questions and essay assignments. It is thus particularly suited for assessing objectives on the levels of Synthesis

Figure 4.17: Screenshot of an experimental function for comparing two submissions to an assignment box.

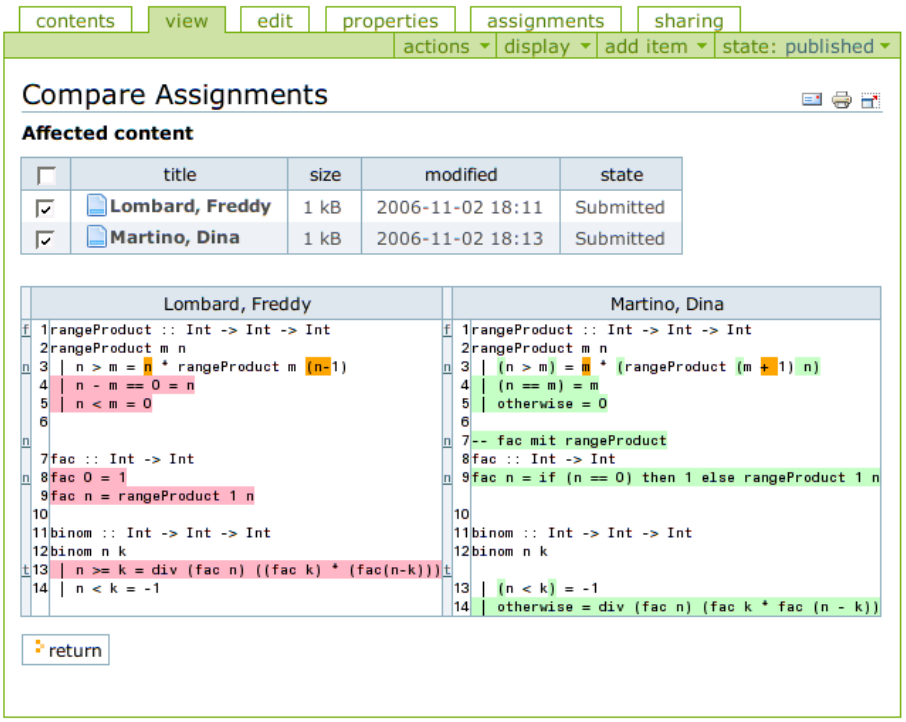
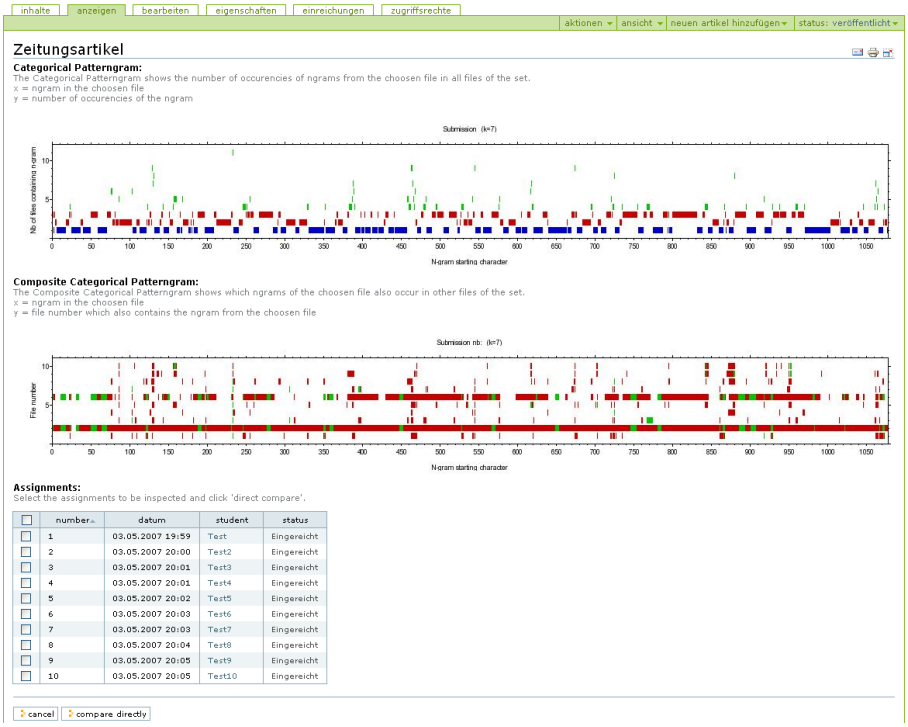


Figure 4.18: The PlagDetector extensions offer a variety of algorithms, tools, and visualizations which can help detecting plagiarism in ECAssignmentBox submissions (screenshot taken from [42]).



and Evaluation, or generally tasks requiring creativity, which are difficult, or, in some cases, impossible, to assess with selected-response questions. Since textual answers and essays cannot be assessed automatically, ECAssignmentBox cannot be used for self assessment, though.

4.4.3 ECAutoAssessmentBox

In computer science, programming is an essential topic in the curriculum. Practice is essential for the acquisition of programming skills, so students should be given frequent programming assignments. However, assessing large number of programming assignments is a time-consuming and labor-intensive task; it is thus attractive to automate it as far as possible. The first systems for automatically testing student programs were developed and used as early as 1965 [see 55].

ECAutoAssessmentBox is a Plone product derived from ECAssignmentBox, i.e., it has the same basic functionality as described in section 4.4.2 above. In addition, ECAutoAssessmentBox provides support for *automatic evaluation* of student submissions.

Automatic evaluation is not implemented in ECAutoAssessmentBox itself but realized in conjunction with a separate Web service, ECSpooler. ECAutoAssessmentBox acts as a front end to ECSpooler: Whereas an assignment box realized with ECAssignmentBox stores its submissions for perusal and assessment by an instructor, an auto-assessment box not only stores the submissions, but also forwards them to ECSpooler for automatic testing.

ECSpooler is a Web service running independently from Plone and manages a submission queue and several *backends*.[28] Each backend provides syntax checking and testing for a specific programming language. At the time of this writing, backends have been implemented for the Common Lisp, Erlang, Haskell, Java, Prolog, Python, and Scheme programming languages; for Haskell and Java there are alternative backends using QuickCheck [32] and JUnit[29] for testing submissions. After running its tests, the backend reports its results back to ECSpooler, which, in turn, will provide them to ECAutoAssessmentBox, which displays them to the student, as shown in figure 4.20 on page 115. Thus, students receive immediate feedback on their submission.

While the system (ECAutoAssessmentBox, ECSpooler, and backends) was primarily intended for programming assignments, with the appropriate backends, it can also be used to check submission in other formal notations, such as regular expressions or mathematical formulas.

Backends can implement any testing strategy desired and need not return a "pass/fail" indication. The backends used at the WDOK research group all use *dynamic analysis*, i.e., the submitted programs are actually executed. However, *static analysis*, such as provided by lint for the C programming language [82] and by similar tools for other languages, could also be utilized in the assessment of student programs.

28. For a detailed description of ECSpooler and its architecture we refer readers to Amelung [2].
29. `http://junit.sourceforge.net/`

Natural-language texts obviously cannot be tested for correctness like computer programs; however, automatic analyses such as those provided by the classic UNIX Writer's Workbench [29], i.e., possibly specialized spelling, grammar, and style checkers, could be used to give feedback to students and help instructors with classifying and grading natural-language submissions, especially when large numbers of essay-like submissions are to be assessed. In a master's thesis supervised by the author, Feustel [53] has implemented an experimental backend providing style checking and keyword spotting.

Creating Assignments with ECAutoAssessmentBox

The creation of auto-assessment boxes is basically identical to the creation of assignment boxes with ECAssignmentBox, as described in section 4.4.2 above, except that some additional information is required for automatic testing. In any case, instructors have to select the backend to be used for testing; depending on the backend and the type of tests it implements, other information may be required, such as test data and a model solution (see figure 4.19 on the next page).

The Students' View

The students' view of an auto-assessment box is at first identical to that of an assignment box (see figure 4.10 on page 102): Students are presented an assignment and can enter their solution into a text field or upload a file.

The difference is that after submitting a solution to a regular assignment box, students only receive a confirmation message, whereas they get automatic feedback after submitting to an auto-assessment box, see figure 4.20 on page 115.

Discussion

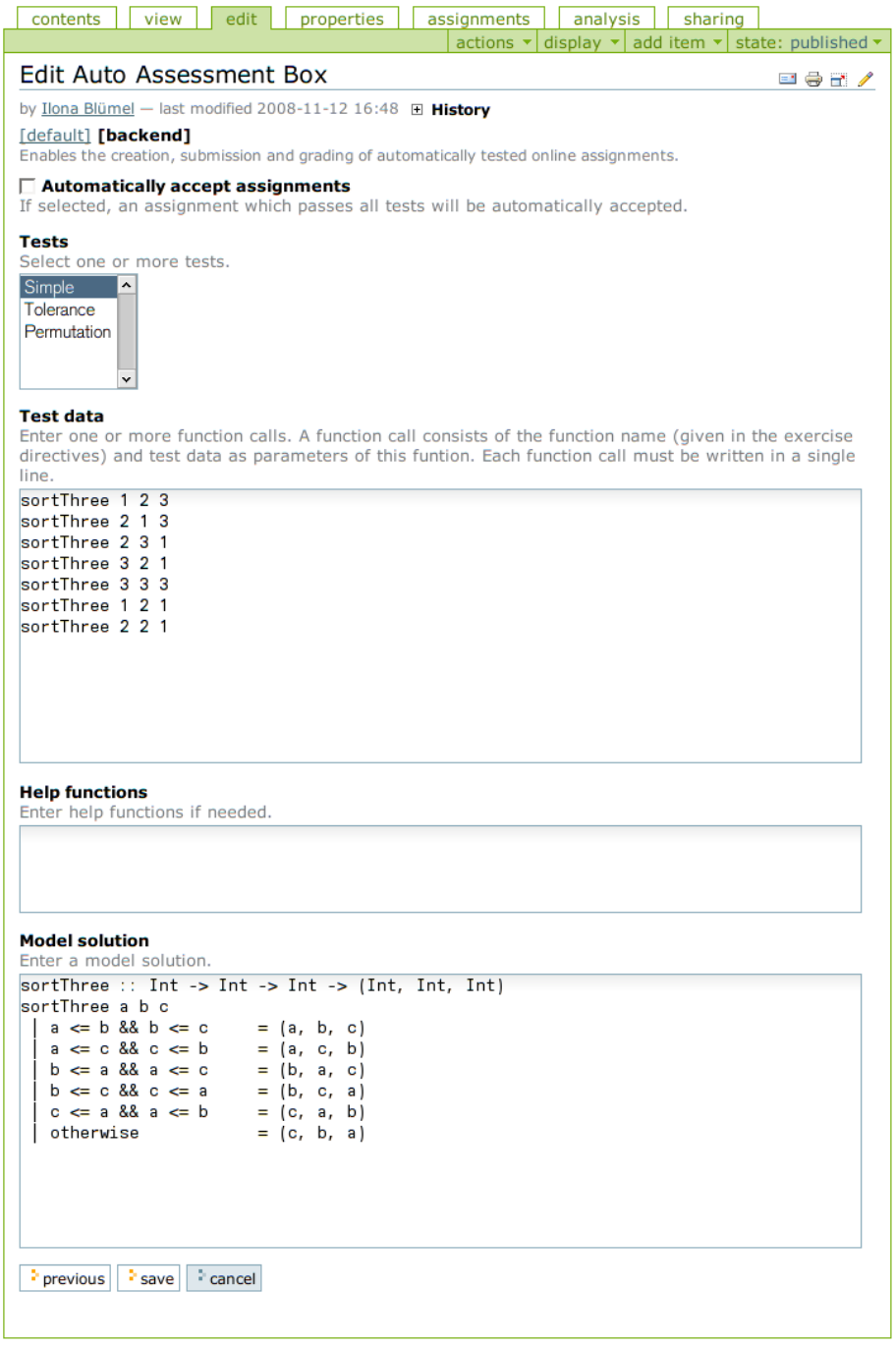
Automatic testing is not the focus of this dissertation; for an extensive discussion of automatic testing in the context of the eduComponents readers are referred to Amelung [2]. However, since ECAutoAssessmentBox is a part of the e-learning environment offered by the eduComponents, we will briefly discuss the motivation for automatic testing in computer science education.

Our motivation for implementing a component for automatic testing was twofold. First, automatic testing reduces the workload of instructors, as it frees them from the tedious job of running the submitted programs themselves. It also supports them in judging the acceptability of program submissions, since they can easily see which programs run and produce the expected results. This allows instructors to assign students more programming exercises, helping them to gain more programming practice.

Second, automatic testing provides students with *immediate* feedback on their programs, which is important to keep them motivated.

Practice is essential for learning programming. In his study of the motivation of students of programming, Jenkins [80] points out:

Figure 4.19: This screenshots shows the backend options for an auto-assessment box using the Haskell backend. This backend runs the student solution on a set of test data and compares the results with those of a model solution; this information must thus be specified by instructors.



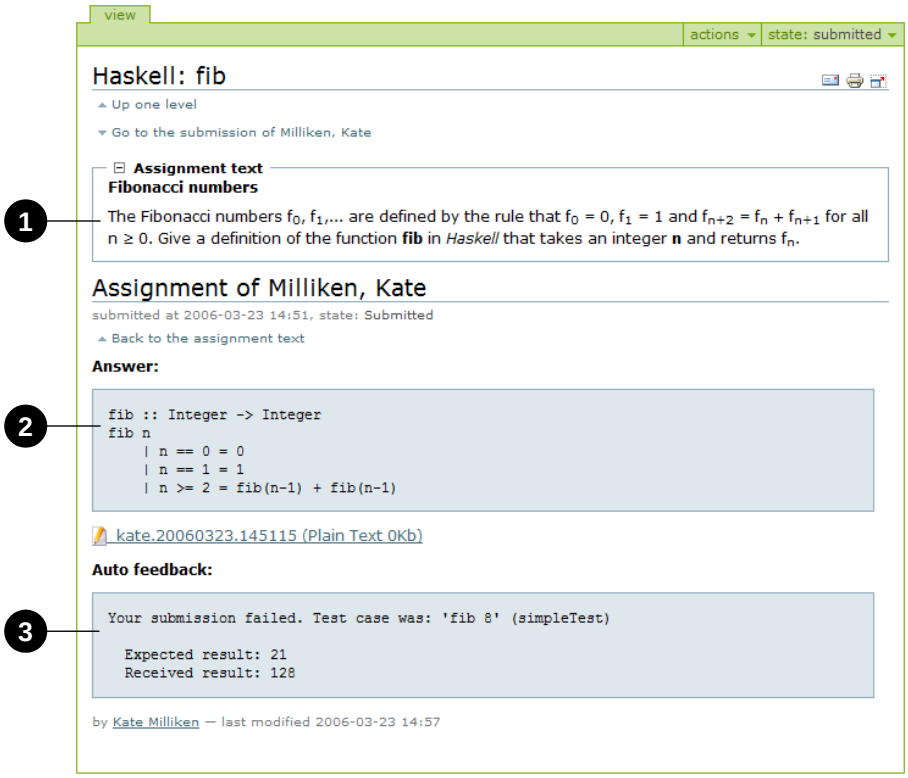


Figure 4.20: ECAutoAssessmentBox automatically tests submissions to programming exercises and immediately offers feedback. This figure shows the view a student gets after the automatic testing of a programming assignment. ❶ is the assignment; ❷ is the student's submitted program, in this case an incorrect solution; ❸ is the automatic feedback, reporting an error, since the submitted solution does not yield the expected results.

> Programming is a difficult skill to acquire. It is best learned by practice and, if students are to learn effectively, some at least of this practice will have to be self-directed. An instructor's key role is to persuade the students to do this and thus to *motivate* them. They must be motivated so that they will engage appropriately. [Emphasis in original] [80, p. 56]

Our experience—and that of others—has shown that quite a number of students tries to avoid writing and testing programs and content themselves with non-working sketches, thus depriving themselves of programming practice. In general, students do not work on exercises voluntarily. For example, in her analysis of student grades and student participation, Cantaluppi [26], p. 78 reports that in all analyzed courses the participation and the grades of students were, on average, better in the first half of the course, where exercises were obligatory or resulted in a bonus, whereas they were voluntary in the second half. Obviously, students can be said to take a "minimalist" approach.

Thus, to ensure that students actually do the exercises, they must be mandatory. For example, utilizing automatic testing, Saikkonen et al. [150] assign "small but mandatory weekly programming exercises in order to keep the students active during the whole course instead of trying to catch on the course just before the exams." In this scenario, automatic testing can also ensure that students remain motivated: The instant feedback provided by automatic testing is also a motivational factor.

It is known that feedback is crucial for learning: "Knowing what you know and don't know focuses learning. Students need appropriate feedback on performance to benefit from courses." [30]. Gibbs and

Simpson [59] identify a number of conditions under which formative assessment, supports students' learning: Feedback plays an important role. However, a decisive factor for feedback to be useful is timeliness: If students receive it too late, they will already be working on a different assignment and it will be very unlikely to have any effect. Thus, as Gibbs and Simpson point out, "imperfect feedback from a fellow student provided almost immediately may have much more impact than more perfect feedback from a tutor four weeks later."

We did not intend automatic testing to replace the testing of programs by students with the appropriate compiler or interpreter. To the contrary, when the number of tries is limited, students must test their programs thoroughly *before* submitting them, which also encourages them to think about design and testing issues. As Douce et al. [47] point out, automatic testing, together with a grading system that awards no points for incomplete solutions, increases the importance of writing completely working solutions in the eyes of the students.

While the feedback provided by our backends may be considered rudimentary (this is in part intentional, since they are not designed as a tutoring system), the immediate feedback was mentioned surprisingly often as very helpful by our students (see section 5.3). This positive reaction to the automatic feedback may be caused by the fact that previously students received feedback for their programming assignments only very rarely, namely when they were called up to present their solution.[30] Thus, even though the automatic feedback may not yet be perfect, it represents a notable improvement for the students' learning experience.

Educational Objectives

Like ECAssignmentBox, ECAutoAssessmentBox is a tool for assessment on the levels of Synthesis and Evaluation through constructed-response questions. The combination of ECAutoAssessmentBox, ECSpooler, and testing backends was specifically designed for automatically testing submissions to programming assignments. Programming assignments can be considered a special case: While programming requires creativity, the outcomes can at least be partially assessed automatically. The currently implemented backends are not specifically designed for self assessment, but ECAutoAssessmentBox can, with the appropriate backends, also be used in self-assessment scenarios.

4.4.4 ECReviewBox

Like ECAutoAssessmentBox, which we discussed in the preceding section, ECReviewBox is a specialized product based on ECAssignmentBox. ECReviewBox inherits the basic functionality of ECAssignmentBox and implements specialized functionality for *peer review assignments* based on submissions to a previous assignment box realized with ECAssignmentBox. *Peer review* here means that students are given anonymized copies of other students' submissions and are asked to review them according to some set of criteria defined by the

30. See section 5.2 for a description of the course structures before the introduction of e-learning.

instructor. Thus peer review assignments work in a way very similar to the peer review of papers submitted to a scientific conference or journal.

Peer Review Assignments

ECReviewBox can be considered an add-on for ECAssignmentBox, allowing instructors to create peer review assignments for submissions to a previous assignment box. The following example illustrates the process:

1. The instructor creates an assignment using ECAssignmentBox or ECAutoAssessmentBox, i.e., the assignment can also be an automatically checked programming assignment.

2. The students submit their essays or programs to this assignment box.

3. The instructor then creates a *review box* using ECReviewBox, which references the original assignment box. Instructors do not have to worry about the assignment of submissions: ECReviewBox randomly assigns a submission from the referenced assignment box to each student, ensuring that nobody gets to review their own submission. Instructors can inspect the assignment of submissions. ECReviewBox also enforces that only students who have made a submission to the original assignment are allowed to participate in the peer review.

4. When students view the review box—which looks and behaves just like a regular assignment box, see figure 4.21 on the next page—they will be presented with a submission to review and with instructions for the review (the peer review assignment). They can then submit their review just like they submit answers to normal assignment boxes.

Since review boxes have all features of regular assignment boxes, the same facilities for handling submissions as in ECAssignmentBox are available.

Educational Objectives

Peer review can be regarded as a prototypical example of Evaluation in Bloom's Taxonomy: Students have to judge the value of material for some purpose or according to some set of criteria and must be able to state why a specific judgment was reached (see section 4.1). In other words:

> Peer assessment has the potential to encourage and enhance critical thinking skills and to help students progress toward a learning model where evidence, rather than the proclamations of an Authority, is the basis for understanding. [31]

Zeller [190] and Sitthiworachart and Joy [166], among others, report noticeable benefits of peer reviews for the learning of programming in

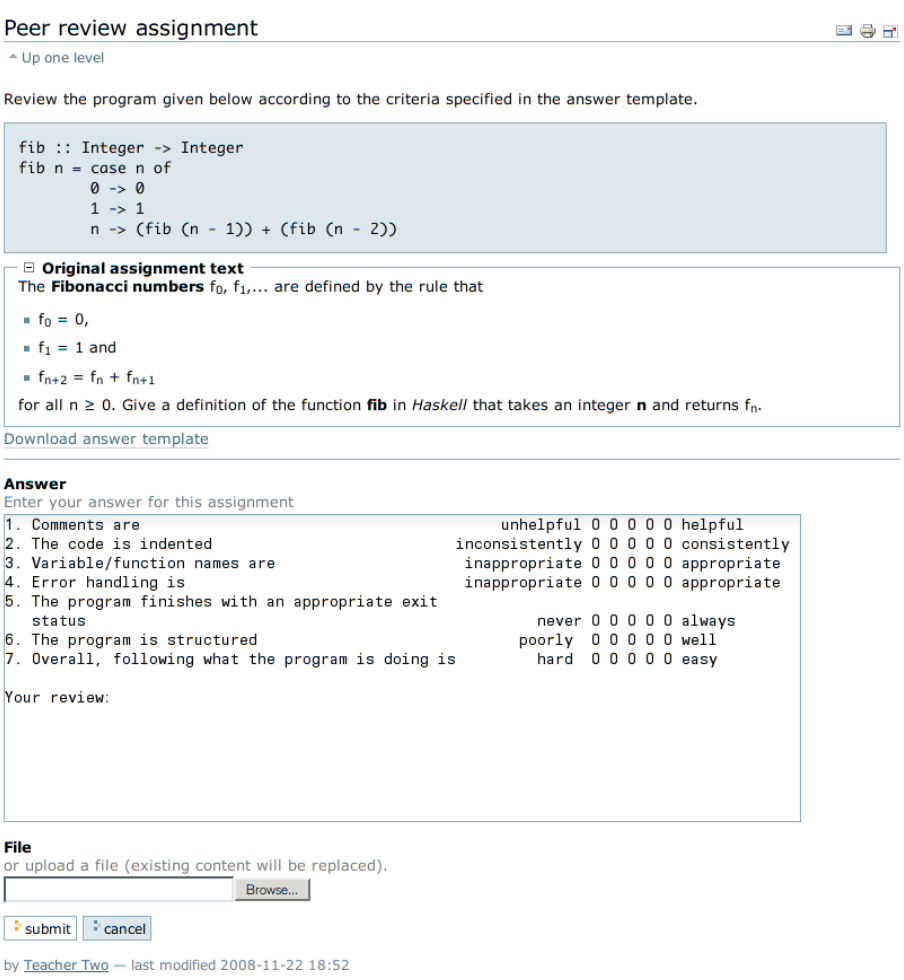


Figure 4.21: This screenshot shows a student's view of an example peer review assignment realized with ECReviewBox. From top to bottom it shows the review assignment, the program to review, the text of the original assignment for which the program was written, and the answer field, pre-filled with an answer template (example rating categories taken from [166, p. 124]) for students to fill out.

computer science education, which has also been the subject ECReviewBox has been used for at WDOK. Of course, peer reviews can also be used for non-programming assignments and in other subjects.

Nevertheless, in her analysis of assessment methods, Reinmann comes to the following conclusion with respect to the use of peer assessment at German universities: "Weder Self- und Peer-Assessment noch formative Assessment-Formen spielen im Lehralltag deutscher Universitäten eine nennenswerte Rolle"[31] [142, p. 18]. Reinmann notes that formative assessment is most often rejected on the basis of time and resource constraints.

To improve teaching and learning, Reinmann argues for more variety in assessment and specifically for newer and better forms of assessment, which include self and peer assessment; she puts forward the thesis that "Web 2.0 technologies" could enable better assessment without the need for additional resources. Two of the factors Reinmann cites to support this claim are the observation that today's student work is frequently *born-digital* and thus easy to collect and process, and that the integration of self and peer assessment into teaching could allow for more and more frequent assessment without imposing an additional burden on instructors.

31. "Neither self and peer assessment nor formative types of assessment play any significant role in day-to-day teaching at German universities."

This is precisely the approach of the eduComponents; with respect to peer review, ECReviewBox takes advantage of the fact that ECAssignmentBox and ECAutoAssessmentBox submissions are digital and makes them available for peer review, seamlessly integrating peer review into the eduComponents environment and enabling instructors to use this interesting teaching device more often.

We would also like to note that peer reviews are clearly a type of assignment which would be very hard to implement without computer support: Even though paper submissions could be copied, it would be difficult and time-consuming to anonymize them (especially in the case of hand-written submissions), to distribute them to the reviewing peers, and to collect and evaluate the reviews. In the case of programming assignments, there would be the additional issue that the programs could not be executed—at least not without transcribing them, which is not only time-consuming but also likely to introduce new errors.

Related Work

Peer review of electronic submissions is not a new idea; Chinn [31] gives numerous references specifically for its uses in computer science. There are also a number of implementations of systems for computer-supported peer reviews in computer science, e.g., Praktomat [190] or BOSS [166, 85]. Some of these systems currently provide more advanced features compared to ECReviewBox; BOSS also offers both automatic testing of programming submissions and plagiarism detection functions.

Educational peer review is also used in other disciplines. An example from the humanities is the COLAC model by Schiltz and Langlotz [154], which uses the scenario of a scientific symposium for teaching how to write academic papers.[32] Participants are split into groups and act alternatingly as authors and reviewers. In the eHistLing project [153], a blended learning approach is taken, where the process is supported by online tools.

However, the focus of this dissertation is not on peer review, so we will not go into further details here. In the context of this dissertation, ECReviewBox may rather be seen an example of how electronic submissions can be reused for new purposes, how new components can be derived from existing components, and how the component-based architecture of the eduComponents allows for new assessment functionality to be added to the e-learning environment.

4.5 Authoring and Interchange of Multiple-Choice Tests

Unlike typical essay assignments, multiple-choice tests are not simple text documents but complex and highly structured entities constructed from numerous parts, namely the indidual test items and their answer choices. A quiz is effectively a program that defines a user interface for

32. The COLAC model was a finalist of the 2006 Medida-Prix (a German-Austrian-Swiss competitive award for e-learning projects organized by the Gesellschaft für Medien in der Wissenschaft).

the candidate and, given input in the form of the candidate's choices, calculates a resultant test score according to some specification.

The authoring and interchange of multiple-choice tests is thus a complex issue which must be considered when developing a quiz system. While this is not the focus of this dissertation, valuable insights were gained during development of the ECQuiz product.

This section is an excursus in which we discuss some of the issues related to authoring and interchange of tests encountered in the design and implementation of ECQuiz. We will first analyze file formats for authoring—in particular ECQuiz's Quick Edit facility and file format—and then examine the IMS QTI standard for the interchange of questions and tests.

4.5.1 File Formats for Authoring

File formats are not only necessary for storing quizzes and for interchanging questions and tests between different delivery platforms, but they also provide a way to interface a test delivery platform to a test authoring system. We use the term *delivery platform* to refer to a system (or part of a system) which is primarily used for making tests available to candidates and to allow candidates to take the tests, whereas *authoring system* refers to a system (or part of a system) primarily designed for creating tests. E-learning platforms with assessment facilities typically provide support for both authoring and delivery of tests; however, authoring support may be limited, so that specialized tools may be preferred by test authors.

ECQuiz also allows both authoring and delivery of test. Nevertheless, we noticed that while using the GUI to create tests in ECQuiz is easy and quite intuitive, it is a relatively slow process because the user has to insert each question and each answer separately. This means that for creating a simple multiple-choice question with four choices the user has to create a question object; then, inside the question object, four answer objects have to be created, and for each object a form has to be filled out.

We also perceived a need for a facility for creating tests offline: The Web interface requires test authors to have a relatively fast Internet connection. There are a number of scenarios where such a connection is not available, for example on a train. It would therefore be useful if tests could be created locally, without the need for an Internet connection, and then, when a connection is available, they could be uploaded to ECQuiz.

MC Frog

To fulfill both requirements—faster test authoring and offline authoring—we decided to create a stand-alone editor for tests, i.e., an editor independent from ECQuiz and Plone. Our rationale was that a local application could be used offline and that a native user interface would be richer and more responsive than a Web interface. This idea was

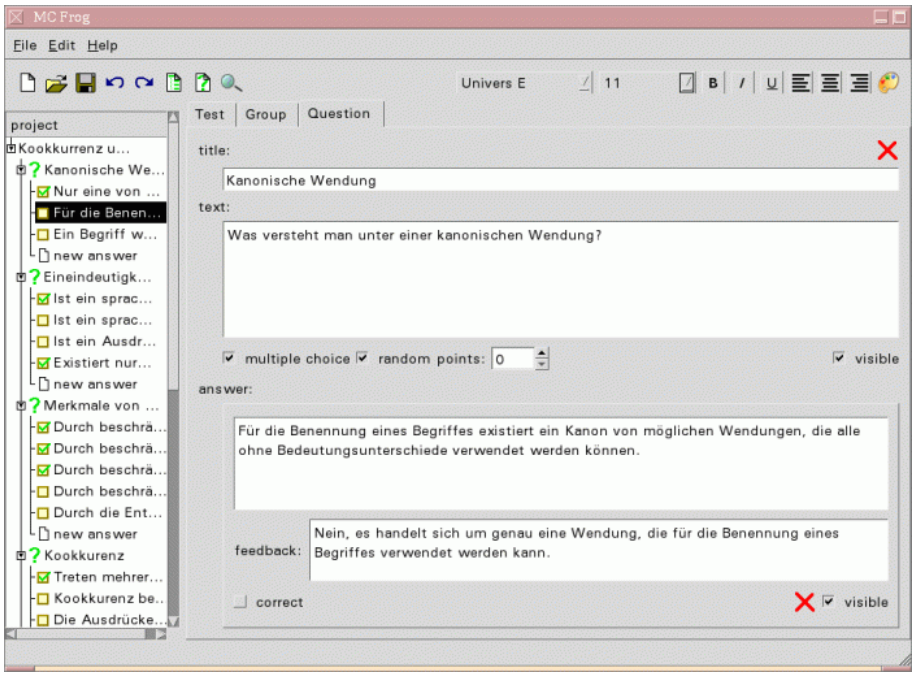


Figure 4.22: Screenshot of the MC Frog test editor.

implemented in the MC Frog test editor by students[33] during a software laboratory course in 2005.[34]

MC Frog provides a graphical user interface for constructing tests (see figure 4.22) and can import and export tests as QTI 2.0 files (packaged as IMS content packages) for use with ECQuiz. According to the developers, the largest hurdle they faced was QTI, even though MC Frog was only required to be interoperable with ECQuiz.

While MC Frog worked according to the specification, we found that creating tests was still quite tedious. We realized that if a test consists primarily of textual material, the problem is that the GUI interferes with the writing process: The user cannot concentrate on writing the items but must switch to navigating the interface after each item, which constitutes a cognitive load for users, distracting them from their proper task. We therefore continued to search for an easier way for creating tests.

Text-Based Formats

Since their invention by Ward Cunningham in 1994 [99], *wikis* have quickly become a widespread way for creating content on the Internet. One important aspect of wikis is that text is written in a non-WYSIWYG fashion using simple markup languages[35]; for example, bulleted lists are typically created by starting a line with an asterisk, and emphasized text is marked up by surrounding it with asterisks, e.g.:

```
Introductory text with a *highlighted term*:

* This item
```

33. André Klonz, Rico Andrich, and Steve Schneider.
34. Software and documentation are available from `http://www.andre-klonz.de/mefir/content/2005/mc-frog/` (accessed 2008-10-17).
35. Some of them are perhaps better called “editing conventions.”

```
* That item

* Another item

Subsequent text.
```

A wiki engine may then render this input as follows:

> Introductory text with a *highlighted term*:
>
> - This item
> - That item
> - Another item
>
> Subsequent text.

The idea behind wiki markup is that it is easy to learn—since it is based on existing conventions or because the markup alludes to the rendered form—and that it requires no special software for reading and writing it, so that as many people as possible can contribute. And in fact, wiki markup is now being used by large numbers of people, e.g., in Wikipedia[36]. This also shows that despite the massive move to user interfaces based on the WIMP (windows, icons, mouse, pull-down menus) and WYSIWYG (what you see is what you get) paradigms since the 1980s, users apparently not only accept, but actually prefer, text-based, non-WYSIWYG interfaces for some tasks.

There are a number of formats for describing tests in a wiki-like syntax[37]. The most widespread wiki-like format is perhaps Moodle's *GIFT* ("General Import Format Technology") format for describing quizzes. For example, a multiple-choice question may be written as:[38]

Listing 4.1: A multiple-choice item in GIFT format.

```
Who's buried in Grant's tomb?{~Grant ~Jefferson =no one}
```

This format is easy to learn and efficient for simple items. For more complex items, however, the syntax is no longer intuitive but more or less arbitrary:

Listing 4.2: A more complex item in GIFT format.

```
::Jesus' hometown::Jesus Christ was from {
   ~Jerusalem#This was an important city, but the wrong answer.
   ~%25%Bethlehem#He was born here, but not raised here.
   ~%50%Galilee#You need to be more specific.
   =Nazareth#Yes! That's right!
}.
```

Other wiki-like formats for describing tests include the "Aiken" format (apparently also a Moodle design, listing 4.3), WebCT's text format (listing 4.4), and Respondus "Standard Format" (listing 4.5).[39]

36. `http://wikipedia.org` (accessed 2008-10-17)
37. Some formats probably predate wikis, nevertheless they are similar to wiki syntaxes—and wiki syntax in turn is also based on earlier conventions from e-mail and Usenet communication.
38. The following GIFT examples (listings 4.1 and 4.2) are taken from the Moodle online help (`http://docs.moodle.org/en/GIFT` (accessed 2008-12-11)).
39. The examples in listings 4.3 and 4.4 are taken from the Moodle documentation (`http://docs.moodle.org/en/Aiken` (accessed 2008-12-11) and `http://docs.moodle.org/en/WebCT_format` (accessed 2008-12-11), resp.).

Listing 4.3: A quiz item in Aiken format.

```
What is the correct answer to this question?
A. Is it this one?
B. Maybe this answer?
C. Possibly this one?
D. Must be this one!
ANSWER: D
```

Listing 4.4: A quiz item in WebCT text format.

```
# Start of question: Multiple Choice Question
:TYPE:MC:1:0:C
:TITLE:Multiple Choice Question
:FEEDBACK
Darwin invented the theory of evolution and created Darwinism.
:QUESTION:H
Where is the Darwin Research Center?
:IMAGE:
:LAYOUT:vertical
:ANSWER1:0:H
Menlo Park, California
:REASON1:H
Sorry!
:ANSWER2:0:H
Vancouver, Canada
:REASON2:H
Sorry!
:ANSWER3:100:H
Galapagos Islands, Ecuador
:REASON3:H
Correct Answer!
:ANSWER4:0:H
London, England
:REASON4:H
Sorry!
:ANSWER5:0:H
Sidney, Australia
:REASON5:H
Sorry!
:CAT:Default
# End of question: Multiple Choice Question
```

Listing 4.5: Quiz item in Respondus Standard Format (example taken from [143]).

```
Title: Speed of Light
3) Who determined the speed of light?

a. Albert Einstein
@ No. Albert Michelson determined the exact speed of light.

*b. Albert Michelson
@ Yes. Albert Michelson won the Nobel Prize for Physics for
determining the exact speed of light.

c. Thomas Edison
@ No. Thomas Edison did not determine the exact speed of light.

d. Guglielmo Marconi
@ No Guglielmo Marconi did not discover the exact speed of
light, but he did win the Nobel Prize for Physics for his work
with radio waves.
```

The different formats share many characteristics. For example, most formats are more or less line-oriented and require rigid adherence to the layout: Typically a new item must start on a new line, various parts are identified by specifics characters at the beginning of a line, and in some cases, e.g., the Aiken format, the question must be all on one line.

Furthermore, some information is only given implicitly; in GIFT format, for example, an item containing *only* true answers (marked with "=") is treated as a short answer question.

These properties and the fact that there are no formal definitions for most of these formats (including GIFT) make them unsuitable for use as interchange formats. We consider them also suboptimal for authoring questions, as the syntax for more complex items is hard to remember, and, as the format is hard to parse, it is difficult to give test authors good feedback in case of errors.

Quick Edit

In ECQuiz 1.1 we therefore implemented an experimental language for describing tests, which is not based on *ad-hoc* conventions but on *S-expressions* [113, 147]. S-expressions, best known from Lisp, are a proven notation for representing complex data in concise, human-readable textual form—or, as Reid argued: "As 25 years of experience with LISP has demonstrated, simple lexical quoting and parenthesis balancing will serve to represent any list structure" [141, p. 116].[40]

As multiple-choice tests can be considered list structures consisting of several test items, each of which consists of a question and a list of answer choices, S-expressions can also be used to represent multiple-choice tests. S-expressions can be used to describe the same structures as XML [see 91] but have less markup overhead, which makes them easier to read and edit without specialized editors.

Listing 4.6 shows an example of a simple item in this format.

Listing 4.6: Item in Quick Edit format.

```
(mcq "Obscure Groucho Marx reference" 1
     "Who's buried in Grant's tomb?"
     (:multsel f)
     (f "Grant")
     (f "Jefferson")
     (t "no one"))
```

This example corresponds to the example in GIFT format shown in listing 4.1 on page 122. The item is represented as a list starting with the keyword `mcq`, indicating that it is a multiple-choice question. The further list elements are the title of the item, the score (here 1), and the question itself ("Who's buried in Grant's tomb?"). This is followed by a sublist of options; in the example only one option, `:multsel`, is given; the value `f` (false) indicates that multiple selection is not allowed, i.e., this is a single-answer item. The options list is followed by the answer choices; each choice is a list starting with either the keyword `t` for correct or the keyword `f` for incorrect answers, followed by the text of the answer choice, e.g., "Grant."

40. As Reid wrote this in 1989, this would now be rather 45 years.

At first glance, the S-expression-based format—currently called *Quick Edit* after the ECQuiz feature—may seem less "natural" than the corresponding GIFT example (listing 4.1 on page 122). However, the Quick Edit format is syntactically much more consistent—since all information is specified in the form of parenthesized lists—and does not require the user to remember the meaning and use of `:`, `~`, `=`, or `#`. GIFT furthermore requires that all of these characters, as well as the braces `{` and `}`, must be escaped by preceding them with a backslash if they are to be used literally as part of question and answer texts.

Inside Quick Edit strings, only `"` must be escaped, analogous to most programming languages. Unlike GIFT, where line breaks have to be entered as `\n`, Quick Edit allows the use of blank lines in questions and answers. Since the various parts of a Quick Edit definition are explicitly delimited (by parentheses and quotes), users are free in the textual layout of the file. As it does not rely on layout and whitespace as GIFT and the other formats described above, Quick Edit is also less prone to undetected errors, i.e., a missing parenthesis in Quick Edit will result in a syntax error, while a missing newline in GIFT may, for example, cause two questions to "run together" without causing an error.

Some systems, e.g., Respondus, also support importing items from CSV-format files[41]. While this format is easy to parse, it is rather inflexible and poorly suited to authoring due to its tabular structure.

Short Informal Description of the Quick Edit Format

Since the design of the Quick Edit format is not yet finished, we will only give a short informal description of its main features here.

The Quick Edit format can describe all item types and item options, and all test options, implemented by ECQuiz, as well as question groups. Listing 4.7 on the next page shows the Quick Edit representation of the quiz shown in figure 4.6 on page 97, illustrating the various item types. Listing 4.7 is in the form as currently generated and exported by ECQuiz and therefore contains some elements (e.g., item descriptions or feedback texts) that are not strictly necessary and which may be omitted when authoring quizzes manually.

A quiz is represented by an S-expression with the keyword `quiz` followed by the quiz title and an optional list of additional specifications, followed by the test items or question groups.

The additional specifications are given in the form of Lisp-style keyword arguments, i.e., pairs of the form `:keyword` *`value`*. Options for quizzes include a description of the quiz (`:desc`), directions (`:dir`), and randomization (`:rand`). Depending on the keyword, the corresponding value may be binary (`t` or `f`), a number, or a string (enclosed in quotation marks).

The structure of quiz items is similar, as they are also represented by S-expressions. The first element is a keyword indicating the type of

41. CSV files are tabular files consisting of lines and columns separated by some delimiter. CSV stands for "comma-separated values," however, the delimiter is often a tab character or semicolon instead. CSV files are typically used for spreadsheet or database data. The format of CSV files is not formally standardized

Listing 4.7: Quick Edit representation of the quiz shown in figure 4.6 on page 97.

```
(quiz "ECQuiz Question Types"
    (:desc "A brief illustration of the question types
            available in ECQuiz."
     :dir "Read the questions carefully."
     :rand f)
    (mcq "MC Question" 10 "Check the correct answer."
        (:desc "" :rand t :randnum -1 :multsel f :tutor f)
        (t "Key" "")
        (f "Distractor" "")
        (f "Another distractor" "")
    )
    (mcq "Question" 10 "Check all correct answers."
        (:desc "" :rand t :randnum -1 :multsel t :tutor f)
        (t "Correct." "")
        (t "Also correct." "")
        (f "Wrong." "")
        (f "False." "")
        (f "Incorrect." "")
    )
    (sq "Scale Question" 10 "Indicate your agreement on the
         following scale."
        (:desc "" :lay h :rand f :randnum -1)
        (0% "Fully disagree" "")
        (33% "Somewhat disagree" "")
        (66% "Somewhat agree" "")
        (100% "Fully agree" "")
    )
    (etq "ET Question" 10 "Write a short answer."
        (:desc "" :templ "It can be said that..." :len 50)
    )
)
```

test item: `mcq` for multiple-choice questions, `sq` for scale questions, or `etq` for extended text questions (see section 4.4.1 for a description of the item types). This is followed by the item title, the item score, and the question text (or *stem*). A list of additional specifications in the format described above may follow; the available options vary somewhat according to the item type and include `:multsel` for multiple-choice questions (`f` for single-answer, `t` for multiple-answer questions), `:rand` and `:randnum` to control randomization, and `:tutor` to indicate tutor-graded questions; scale questions can have an option `:lay` indicating horizontal or vertical layout, whereas for extended text questions an answer template (`:templ`) and the maximum length of answers (`:len`) can be specified. See "Creating Tests with ECQuiz" on page 94 for a detailed description of all options.

The answer choices are also specified in the form of S-expressions. The first element is `t` or `f` for correct or incorrect multiple-choice answers or a percentage for scale question choices. Extended text questions obviously do not have answer choices. The second element is the text of the answer choice, optionally followed by feedback text to be shown when a candidate selects this choice.

Quick Edit Summary

Quick Edit enables test authors to quickly edit a complete quiz in a well-defined[42], concise textual form: Selecting the "Quick Edit" tab of a test displays it in S-expression format for editing. When the "update" button is pressed, the textual form is validated and, if valid, the test is updated to reflect the changes. Figure 4.23 on the next page shows a screenshot of the Quick Edit interface.

Since the Quick Edit format is based on S-expressions, it is more flexible than CSV, easier to process than the GIFT format, and less verbose than XML formats.

ECQuiz also supports uploading and downloading of tests in Quick Edit format (in addition to QTI). This allows authors to create or edit tests offline in a text editor or to use scripts for batch processing. While this is theoretically also possible using QTI, the complexity of QTI and its many problems (see section 4.5.2) make it impractical.

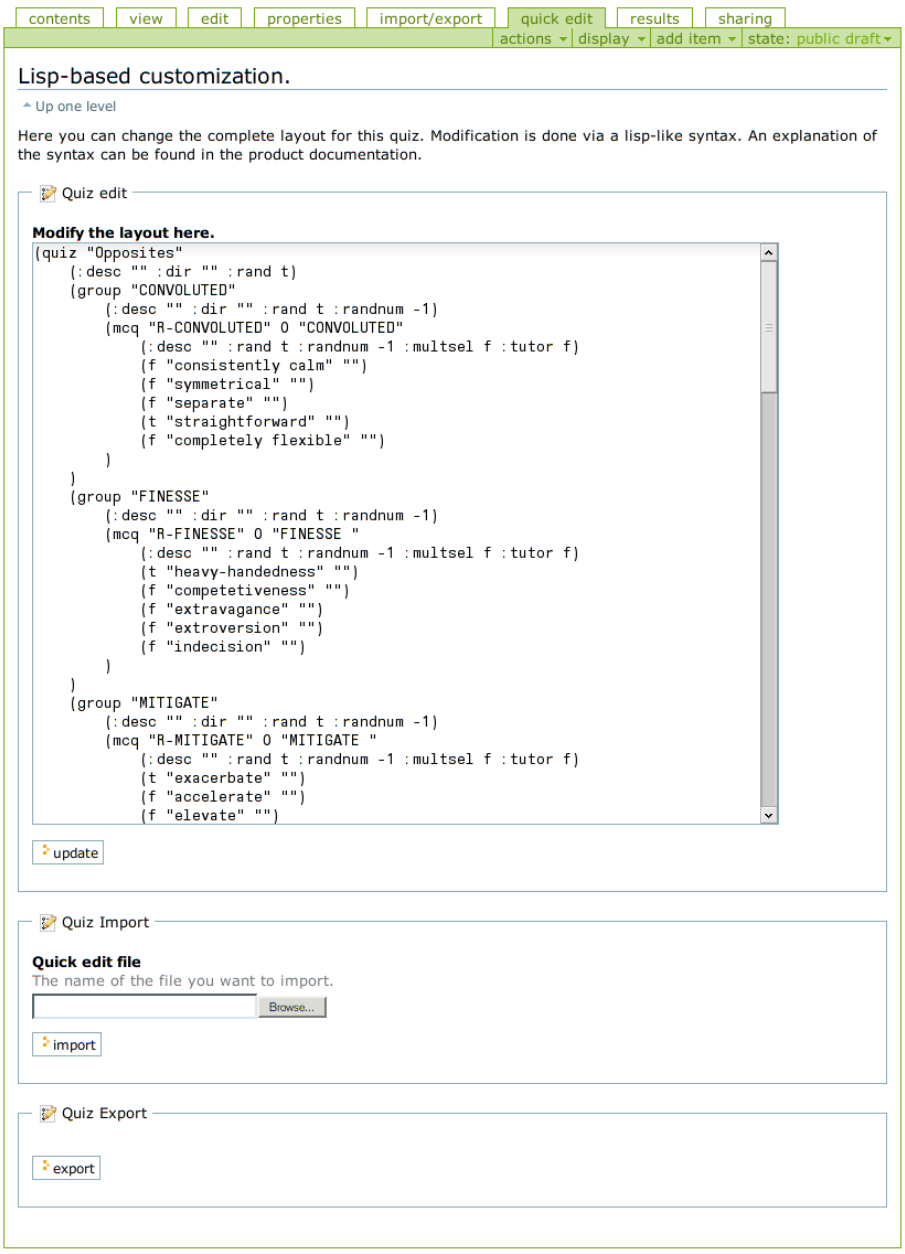
The Quick Edit functionality, including the file format, were introduced as an experimental feature in ECQuiz 1.1. Judging from posts to the eduComponents mailing list, ECQuiz users appreciate this feature and are making use of it.

4.5.2 Test Interoperability[43]

Multiple-choice tests have a number of practical advantages; in particular, scoring can be automated. Creating high-quality multiple-choice tests, however, is challenging, especially if they are to assess higher-order cognitive levels. Or, as Astin [7] puts it:

42. Since the design is not yet finished, we have not given a formal definition here.
43. Portions of the work described in this section have been published previously in [137] (in German).

Figure 4.23: Screenshot of the Quick Edit view of a test in ECQuiz.



> While multiple-choice tests are indeed inexpensive to score, they are extremely expensive to construct: item writing is a highly refined and time-consuming art, especially if one expects to develop good items that are relatively unambiguous.
>
> [7, p. 148]

Ensuring the reusability, longevity, and platform independence of tests can mitigate the high costs of creation and can help preserve investments and intellectual assets when hardware and software change.

Recognizing this, we wanted ECQuiz to be as interoperable with other systems as possible. IMS Question & Test Interoperability Specification (QTI) [71] is currently the only public, implementation-independent specification for the description of multiple-choice tests and similar test types. What is more, the IMS consortium can be considered a *de facto* standards body in the e-learning domain.

QTI conformance was thus a key requirement in the design of ECQuiz from the start.

This section gives a brief overview of QTI and of the implementation of QTI in ECQuiz. Based on our experience with QTI we subsequently present requirements for interchange formats and analyze the suitability of QTI for the purpose of question and test interchange.

A Brief Overview of QTI

The QTI specification describes a data model and a corresponding XML representation for coding assessment items and tests. The declared goal of the specification is to enable the "exchange of this item, assessment and results data between authoring tools, item banks, learning systems and assessment delivery systems" [71, p. 3]. Thus, QTI is intended to be an interchange format.

QTI version 1.0 was published in 2000 and subsequently revised several times. When systems claim "QTI support" (e.g., Respondus, WebCT, or OLAT), this typically refers to a version of QTI 1.x. Smythe and Roberts [168] offer a brief description of QTI 1.0 by members of the QTI working group.

During practical use a number of conceptual problems with QTI 1.x were detected; García-Robles et al. [57], for example, discuss problems that constrain the design of assessment scenarios and limit reusability. The QTI working group therefore considered a completely new design necessary to resolve these issues. QTI 2.0, published in 2005, represents this new design. Despite the promises of far-reaching compatibility by IMS—"software that is compliant with the V1.0 DTD will be able to import V2.0 Items providing it ignores the optional tags" [168]—QTI 2.0 is based on a fundamentally different model and uses a completely different XML structure and is incompatible with QTI 1.x. Furthermore, QTI 2.0 does not cover all areas covered by QTI 1.x; for example, QTI 2.0 can only be used to describe individual items but not complete tests. All further references will be to QTI 2.0 as this is the current official version of the specification.[44]

44. At the time of this writing, QTI 2.1 is still in "public draft" status.

Listing 4.8: Simple example of a QTI 2.0 file.

```
<?xml version="1.0"?>
<!DOCTYPE assessmentItem SYSTEM "imsqti_v2p0.dtd">

<assessmentItem identifier="EX1" title="Formel-1"
                adaptive="false" timeDependent="false">
 <responseDeclaration identifier="REX2" cardinality="multiple">
  <correctResponse>
   <value>choice1</value>
   <value>choice2</value>
  </correctResponse>
 </responseDeclaration>

 <itemBody>
  <choiceInteraction responseIdentifier="REX2"
                     shuffle="true" maxChoices="0">
   <prompt>Indicate the years in which Michael Schumacher was
    Formula 1 racing champion.</prompt>
   <simpleChoice identifier="choice1">1994</simpleChoice>
   <simpleChoice identifier="choice2">2000</simpleChoice>
   <simpleChoice identifier="choice3">2006</simpleChoice>
  </choiceInteraction>
 </itemBody>
</assessmentItem>
```

The QTI specification consists of several parts. The Information Model describes an abstract data model, defining, for example, what a *question* is and what attributes it has. The XML Binding defines a mapping of the abstract model into a concrete XML representation, which is in turn described by a W3C XML Schema and a DTD. Other parts of the specification cover various details and give guidance for implementers and users of QTI.

The basic element of QTI 2.0 is the *item*, i.e., a question and the corresponding answer choices. For the markup of the item content a subset of XHTML [188] extended with test-specific elements is used.

Listing 4.8 is an example of a basic item coded in QTI 2.0; it defines a multiple-choice question with multiple selection. The <itemBody> element contains the question (in the <prompt> element) and the answer choices. Since candidates "interact" with the presented choices, QTI refers to this part as *interaction*. The example uses <choiceInteraction>, with the choices contained in <simpleChoice> elements. The correct choices are specified in the <correctResponse> at the top of the file.

QTI in ECQuiz

As mentioned above, we selected QTI as an interchange format for ECQuiz since we considered interchange of tests important and QTI seemed to be the best—and only—standard available.

However, we did not design ECQuiz in terms of QTI because

- QTI 2.0 had not yet been finalized at the time we started
- QTI is an interchange specification, not a design specification

- Due to the large number of optional items and ambiguities it is not suitable as a design specification.

We therefore first implemented the functionality we needed using a suitable model, and then started to specify a mapping to and from QTI.

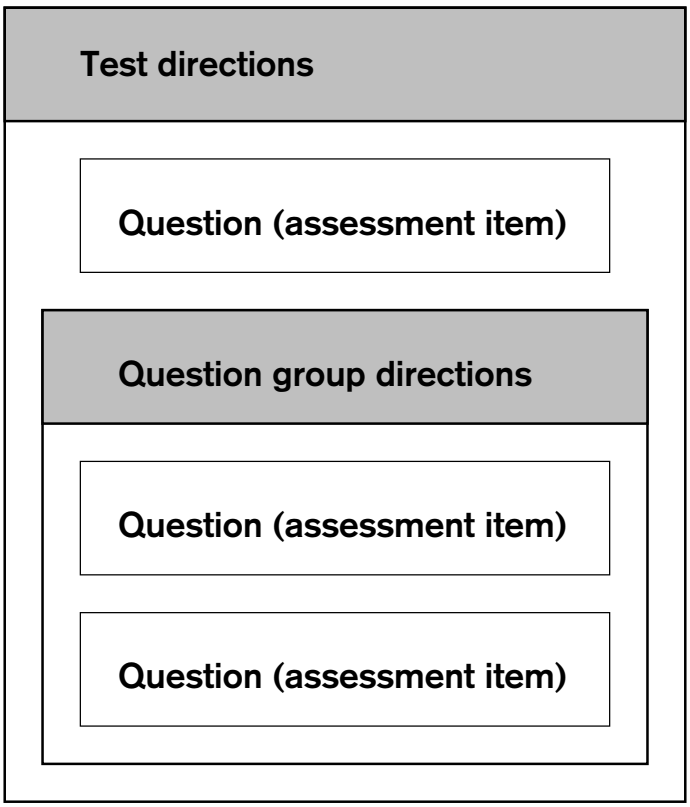


Figure 4.24: Example test structure in ECQuiz. Questions can be grouped into question groups. Question groups and tests can be preceded by prefatory material called directions.

The smallest element in the ECQuiz test model is the *question*. Currently ECQuiz supports two basic types of questions: *Multiple-choice questions*, where candidates have to select one or more answers from a choice of answers, and *extended text questions*, where candidates are supposed to write a textual answer. Related questions, e.g., referring to a common passage or text or an image, can be grouped together in a *question group*; common content is stored in the so-called *directions* of the question group. Figure 4.24 schematically shows an exemplary test structure; figure 4.25 on the next page shows how question groups are displayed.

For ECQuiz we have implented QTI as far as necessary to enable a *round trip*, i.e., we ensured that data exported by ECQuiz could be imported by ECQuiz without loss of information. While designing the mapping from the ECQuiz test model to QTI we encountered several problems that required ECQuiz-specific extensions.

As mentioned above, unlike QTI 1.x, QTI 2.0 specifies only individual assessment items and does not update those parts of the specification that dealt with the aggregation of items into sections and assessments or the reporting of results.[45] Since ECQuiz handles both complete tests and groups of items, it was clear that we needed a way to describe them in a portable way. The QTI Integration Guide and the QTI Migration

45. QTI 2.1 (at the time of this writing in "public draft" status) will add these specifications.

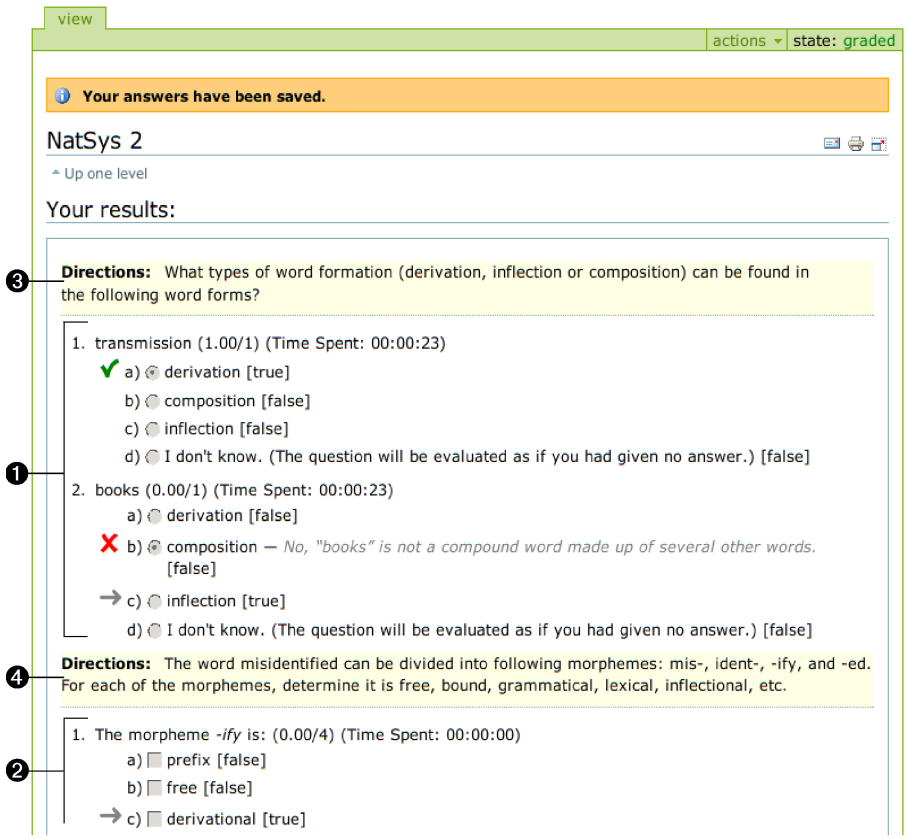


Figure 4.25: Example of a test (in results view) with two question groups (❶ and ❷) and the corresponding directions (❸ and ❹).

Guide briefly mention some relevant issues but many questions with regard to a concrete implementation remain open, for example:

> As this version of the QTI specification does not define either an information model or a binding for section, assessment and objectbank objects no recommendations on how to interpret collections of packaged version 2 items are made. However, packaged items may be referred to individually in an associated learning design or set of sequencing rules. [71, Integration Guide, p. 4]

The Integration Guide also reveals that the integration of the various IMS specifications is far from optimal:

> IMS Learning Design and IMS QTI are natural partners in the learning process. [...] However, the type systems used in IMS LD and IMS QTI differ: [...] A final complicating factor is the presence of multi-valued variables in QTI which have no equivalent in IMS LD. [71, Integration Guide, pp. 7 and 9]

For implementing QTI support in ECQuiz we have chosen the following approach, which involves three IMS specifications:

1. IMS Question & Test Interoperability Specification (QTI) [71]

   Each question (with its associated answers) is mapped to an <assessmentItem>, and thus to a separate file. The prefatory material of tests and question groups is treated <assessmentItem>

without interaction.[46] This approach allows a syntactically valid representation of ECQuiz tests; it is not guaranteed, however, that other systems are able to correctly interpret this use of the <assessmentItem> element.

2. IMS Content Packaging Specification (CP) [70]

   Content Packaging is used to assemble the individual assessment items into a test. *Packaging* basically means that all files, along with a *manifest*, are packed into a ZIP archive. The manifest is an XML file named `imsmanifest.xml` in the root directory of the archive listing the resources contained in the package. Also, the manifest can contain a definition[47] of the structure of the packaged learning material in the <organization> element. This provides a way to represent question groups.

3. IMS Simple Sequencing Specification [72]

   The randomization of answers inside an assessment item is covered by QTI, but ECQuiz also supports the randomization of questions (including the presentation of only a selection of all available questions); this behavior can also be controlled inside question groups.

   The Simple sequencing specification defines elements which can be used to describe the sequential structure of a learning experience. These elements can be used inside the manifest's <organization> elements. In our context, we can use them to specify randomization of questions, number of attempts and availability dates.

Summary The implementation effort required for QTI support in ECQuiz proved to be very high. Since we wanted to ensure that round trips are possible, we had to implement, besides QTI, parts of IMS CP and IMS Simple Sequencing. During the work it became obvious that the specifications are not perfectly aligned. Furthermore, we had to find work-arounds for idiosyncrasies and restrictions imposed by QTI. Consequently, the QTI module accounts for almost 50% of the total code of ECQuiz. To limit the implementation costs and due to a dearth of other QTI 2.0 implementations, the QTI import facility of ECQuiz is primarily designed for the import of ECQuiz-generated items and content packages. Experiments with QTI files and content packages from other sources were unsatisfactory, except for very simple items. For example, QTI 2.0 items and content packages produced by Moodle did not conform to the specifications, so that import into ECQuiz was not possible.

Due to the numerous problems we encountered with QTI, we will discuss them in some more detail in the following section.

46. Theoretically, XHTML could be used for these items, but fragments, e.g., just a <p> element, would not be valid, while valid XHTML documents would require further elements (e.g., <head>, <title>, <body>) which are meaningless in this context.
47. Actually, the manifest can contain more than one structure definition.

Requirements for Interchange Formats

Before taking a closer look at QTI, we should step back and think about the requirements for interchange formats. Whether or not two systems can directly exchange data depends on the existence of a common data format. In practice, however, this alone is not yet a guarantee for trouble-free interchange. The reliability of the interchange depends almost directly on the *specification* of the interchange format. If the interchange format is insufficiently specified it is possible—and even likely—that different developers interpret and implement the specification in different ways. This means that data interchange may be impossible even though all variants may in principle conformant with the specification. Furthermore, it can be assumed that the higher the complexity of the specification, the higher the likelihood for errors in the implementation.

If a system reads or writes the interchange format incorrectly, it is consequently probable that the interchange will not work or will not work correctly. The latter case is potentially more dangerous since the interchange may seem to be successful but, in fact, data may have been lost or corrupted. This type of errors can remain undetected for a long time and may, in the case of tests, lead to candidates being assigned an incorrect score.

Formal Definition These considerations suggest a number of requirements for interchange formats for tests and their specifications. First of all, a standard—or a specification intended as such—should rely as far as possible on existing and proven standards. Many, if not most, specifications for data formats therefore currently are based on XML [186]. This allows keeping the specification of the interchange format concise and enables implementers to make use of available and proven tools.

The use of XML in a specification is, however, only useful if it includes a formal definition in a schema language such as Relax NG [77] or W3C XML Schema [187]. A specification should make full use of the facilities provided by the schema language so that conformance can be verified as far as possible by an XML parser; natural-language requirements and restrictions are hard to verify automatically and thus error-prone and should thus be avoided as far as possible.

A schema describes the *syntax* of a data format. An implementation that is supposed to read and write the format also requires a complete specification of the *semantics*, i.e., the meaning of the individual elements, so that it can interprete and transfer tests in the form intended by the original author.

Separation of Content and Form There are different scenarios for the interchange of tests. In some cases, tests may be reused in their entirety, whereas in other cases only individual items from a test or a test collection (*item bank*) may be integrated into another test. It is therefore essential to clearly separate the different aspects of tests, especially *content*, *appearance*, and *behavior*.

Size and Scope An interchange format that is able to describe all of the functionality of *all* systems may seem desireable to enable the interchange of complete tests with all their properties. Looking closer, one can see that this requirement is illusionary: The facilities of test systems are too diverse and too varied and no system supports all test and scoring types.

An interchange format should therefore restrict itself to a relatively small "core set" of test and item types. Further types can be added later on once it has become clear which types are actually required in practice. The specification of optional parts or alternatives should, however, be avoided at all costs, as it has been shown to severely impede the development of interoperable implementations.[48]

Longevity Finally, an interchange format should ensure longevity, i.e., items and tests described using the format should remain processable for as long as possible. For correcting errors and extending the format new revisions will become necessary from time to time, but it must be avoided that new revisions interfere with the data interchange. This means that different revisions should be compatible with each other as far as possible. Gratuitous incompatibilities must be avoided at all costs; sometimes, however, incompatible changes may be necessary.

To mitigate the potential negative impact of such changes, rigorous revision management is essential, starting with the distinction between major and minor revisions and corresponding numbering schemes. Incompatible changes must then only be introduced in major revisions, after having been anounced before. Inside a major revision, say, 1.x, all minor revisions (e.g., 1.1, 1.2, etc.) are all compatible with each other. Revision 2.0 may introduce incompatible changes, but not 2.1. This ensures that implementors and users can easily and reliably decide whether a specific file can be processed or not.

Problems With QTI

Feedback from users of QTI 1.x was taken into account during the design of QTI 2.0. Nevertheless, we discovered numerous flaws while implementing QTI support in ECQuiz (see section 4.5.2). Most of the problems can be classified into one of three categories: (1) Design problems, (2) formal weaknesses, and (3) technical problems related to the XML mapping.

Note that QTI 2.0 does *not* codify existing practice but was written "from scratch." In contrast to the IETF standards process for RFCs [22], for example, IMS does not require two independently developed interoperable implementations, nor is there a reference implementation for QTI 2.0.

48. The SGML and XML specifications may serve as a good example. The SGML standard [78] contains numerous optional parts: Despite being published over twenty years ago, there is no parser that implements all parts of the standard. XML is a subset of SGML, which does not contain any optional parts: In a very short period of time, a large number of conforming and interoperable implementations have become available and XML has spread almost universally.

Design Problems

QTI 2.0 is a very large specification with many optional parts. To ensure interoperability between systems that do not implement all parts of the standard, the specification provides for the definition of *profiles*. Profiles provide for a way to describe the subset implemented by a system. Two profiles, *QTI-Lite* and *QTI-All*, are predefined.

Both of the predefined profiles are of little practical use. QTI-Lite defines a minimal subset that is too restricted for even the simplest tests: For example, QTI-Lite-conformant items cannot contain enumerations or tables and may only use JPEG and GIF images, but not the W3C-standard PNG format—restrictions which are hard to justify on technical grounds. QTI-All, on the other hand, requires the implementation of the complete specification. At the time of this writing, we are not aware of any complete implementation of QTI 2.0; considering the size and scope of the specification and the problems discussed below, we doubt that a QTI-All implementation will ever be produced.

The QTI specification tends to be quite liberal in many points; for example, almost arbitrary structures are allowed inside an item. This means that an empty item is fully conformant, as are items without <itemBody> element (and thus without a question text) or items with multiple interactions, e.g., an item that is a multiple-choice and a cloze question at the same time.

The advantage of this approach is that many question types can be modeled in QTI. The disadvantage is, however, that the import of QTI items from unknown sources becomes very complex. Since the specification does not define the *meaning* of, say, an empty item or an item with multiple interactions, it is hard to guarantee that an item is imported and interpreted as originally intended by its author. This, in turn, means that the main purpose of an interchange format is not met.

In addition to the description of assessment items, QTI specifies a programming language for *response processing* (RP), i.e, the processing of candidate responses. Since the description of an item and the scoring of candidate responses to an item are two completely different issues, we would argue that the response processing should have better be described in a separate standard. This would have reduced the size and complexity of the QTI specification and may have allowed for a better design of the RP language.

The above-mentioned separation of content, form, and behavior is hardly to be found in the design of QTI. Many element definitions that describe the content of an item, e.g., <feedbackBlock>, <feedbackInline>, or <responseDeclaration>, contain dependencies to the item's behavior, which is defined by the response processing. Even if scoring is not done in terms of QTI response processing (which is legal) specific elements and attributes must always be present for conformance with the specification.

Formal Weaknesses

The QTI specification is often imprecise or ambiguous; many questions remain unanswered and the reader is required to guess. It thus does

not fulfill our desideratum of preciseness, and we find it unlikely for two QTI implementations to agree in their interpretation of the specification to the extent that is necessary for interchange.

For example, the XML Binding defines the data type `language` simply as a "trivial restriction of xsd:string" [71, XML Binding, p. 52]. There is no mention of the value range of the data type.

The definition of the format of the type `identifier`, on the other hand, is very verbose, but equally puzzling. Instead of a formal definition in Backus Naur Form or as a regular expression, the QTI specification gives the following long-winded definition in natural language:

> An identifier is a string of characters that must start with a Letter or an underscore ('_') and contain only Letters, underscores, hyphens ('-'), period ('.', a.k.a. full-stop), Digits, CombiningChars and Extenders. Identifiers containing the period character are reserved for future use. The character classes Letter, Digit, CombiningChar and Extender are defined in the Extensible Markup Language (XML) 1.0 (Second Edition) [XML]. Note particularly that identifiers may not contain the colon (':') character. Identifiers should have no more than 32 characters. for [*sic*] compatibility with version 1 They [*sic*] are always compared case-sensitively.
>
> [71, Information Model, p. 52]

The typos make the reference to "compatibility with version 1" unclear. In contrast to the definition cited above, the XML Schema defines the type `identifier` as `NMTOKEN`, which means that the Schema *allows* periods and colons in identifiers. This, in turn, means that the application has to implement the restrictions given in the specification even though it would have been trivial to implement them in the XML Schema. The specification also does not define whether identifiers *may* be longer than 32 characters and up to which character they have to be unique; in fact, the specification does not contain *any* statement on the uniqueness of identifiers.

Yet another example can be found in the definition of the element intended for free-text answers, <extendedTextInteraction>. This element has the attributes `expectedLines` and `expectedLength`. Both are meant to give candidates some indication of the expected length of their answers [see 71, Information Model, p. 29]:

> **Attribute:** expectedLines [0..1]: integer
>
> The expectedLines attribute provides a hint to the candidate as to the expected number of lines of input required. A Delivery Engine should use the value of this attribute to set the size of the response box, where applicable.
>
> **Attribute:** expectedLength [0..1]: integer
>
> The expectedLength attribute provides a hint to the candidate as to the expected overall length of the desired response. A Delivery Engine should use the value of this attribute to set the size of the response box, where applicable.

It is impossible to determine the difference (if any) between these two attributes from the nearly identical descriptions. No precedence is

defined for the case that both are specified. Furthermore, the meaning of the values is not defined: The specification of a number of lines using `expectedLines` would only make sense if the length of the lines were known; the value of `expectedLength` may, e.g., refer to the number of words, the number of sentences, or even to the width (in centimeters) of the input form. For an application importing an item from an unknown source it is thus impossible to determine the intended meaning of these attributes.

Technical Problems Related to the XML Mapping

As mentioned above, one part of the QTI specification is the XML Binding, defining a mapping of the Information Model to a W3C XML Schema. Regrettably, the facilities offered by XML and XML Schema are only utilized to a small extent, so that QTI files can only be partially validated by standard XML tools. Thus, another one of our desiderata is not fulfilled. We will show examples to illustrate some of the issues.

In many cases, QTI uses an attribute containing the identifier of the target element for cross references. Sometimes, however, other mechanisms are used. As can be seen in listing 4.8 on page 130, the correct choices for a multiple-choice question are indicated using the <correctResponse> element, which contains one or more <value> elements. The *content* of each <value> element is the identifier of a correct <simpleChoice> element.

The problem here is that, if the capabilities of XML had been fully used, a much more robust and elegant solution would have been possible. For cross referencing of elements XML specifically offers the attribute types `ID`, `IDREF`, and `IDREFS`. When using attributes of these types, an XML parser can ensure that all identifiers of type `ID` are unique and that all elements referenced by `IDREF` or `IDREFS` actually exist. It is likely that these are the semantics intended by the QTI specification; however, the QTI XML Schema does not use *any* of these attribute types *at all*.

The element <rubricBlock> is representative for many other shortcomings in the XML Binding. In the definition of this element the Information Model notes: "Although rubric blocks are defined as simpleBlocks they must not contain interactions." [71, Information Model, p. 23] The XML Schema, however, does not enforce this restriction, even though W3C XML Schema provides the necessary facilities to do so.

A Critical Review of QTI

On the basis of our experience with QTI 2.0 during the implementation of a subset in ECQuiz and through the analysis of the QTI specification, our conclusion is that QTI 2.0 is unsuitable for the exchange of tests. While it does, in principle, allow the description of a large number of item types and scoring methods, a complete implementation would require a prohibitive effort, whereas partial implementations—such as the one in ECQuiz—do not achieve the required level of interoperability.

At the time of this writing, QTI 2.1 is being prepared. This version is supposed to add the facilities for describing complete tests, which

are missing in QTI 2.0. Apart from this addition, the current drafts contain no fundamental changes from QTI 2.0, so that our criticism still applies. However, QTI 2.1 contain a number of smaller changes which cause QTI 2.1 to be not completely backward-compatible with QTI 2.0. This means that items conformant with QTI 2.0 will require changes to be conformant with QTI 2.1.

We are not alone in our criticism of QTI. Gorissen has evaluated the QTI support of a number of systems[49] in 2003 and 2006 [61, 62]. His results show that all systems support only very small subsets of QTI. Moreover, in most cases information is lost during import. He also notes that the situation has not improved in 2006.

Strobbe [170] has examined QTI specifically with regard to accessibility, i.e., support for users with disabilities. He reports that QTI 2.0 solves some of the accessiblity issues of QTI 1.2, but that the ambiguity with regard to the intent of interaction types was not sufficiently addressed. Instead of a learning outcome, only the visual renderings are considered; for example, the names of question types such as <hotspotInteraction> or <drawingInteraction> suggest a specific rendering rather than a learning outcome. This problem is compounded by the general lack of well-defined semantics that also hampers effective interchange.

Sclater [157], p. 70 agrees that the QTI specification is "unreadable by the vast majority of candidates likely to be undertaking an online assessment and of no concern to those setting the questions."

The criticism of QTI brings up a number of questions. Gorissen [62] states that there is an "obvious business need for better interoperability." The question is then why we still have the current state of low interoperability. Gorissen's conclusion is that "the market does not yet see the importance of that and their customers fail to explain that need to them." To remedy this situation, he calls for more explicit action from funding bodies, such as requiring the use of tools that support interoperability. Regarding the complexity of QTI, which hinders the development of QTI support, Gorissen suggests the development of an open-source reference implementation by the "educational community," so that the effort for implementing QTI support is reduced.

Sclater [157] concurs that "IMS QTI is unarguably a complex and difficult specification for vendors to implement," but he argues that this is not the primary reason for the slow adoption of QTI. Like Gorissen, he finds that the market does not demand interoperability, however for different reasons: Universities own the assessment process and thus have no need to exchange assessment data with other institutions, so that interoperability is effectively irrelevant. Furthermore, commercial vendors are largely uninterested in interoperability, since they see no clear business case for it, and even if interoperability is a (nominal) selection criteria, institutions tend to simply believe vendors' claims, so that there is little pressure on vendors to invest in interoperability.

49. The systems reviewed in 2006 were: Respondus, Questionmark Perception, N@tschool!, Blackboard, Learn eXact (QTI 1.2), and TOIA (QTI 2.1)

Furthermore, Sclater sees the market as basically split up between "BlackCT"[50] and Moodle, and since the systems take radically different approaches, he considers it unlikely that an institution which has opted for one system will ever move to the other system, though he acknowledges that a growing number of universities are moving from commercial e-learning platforms to Moodle.

His conclusion is that there is currently effectively no *need* for interchange formats and only a market for e-assessment *content* would drive QTI adoption.

Sclater's comments and conclusions should perhaps be seen on the background of an earlier publication, Sclater et al. [158], a 2002 report of an interoperability evaluation for QTI 1.1 and 1.2.[51] The results of this evaluation were—like those reported by Gorissen [61]—disappointing: Both LMSs effectively failed to either import or export QTI-encoded tests; the other systems had various problems, including showing the wrong feedback text, even though the two test items used in the evaluation were extremely simple. In the light of these results Sclater et al. thus wondered:

> If, despite the programmers' expertise and the relative simplicity of the questions chosen, they are still failing to be rendered correctly does this bode well for interoperability?
>
> [158, p. 324]

Practically the same comments were made in 2002 regarding the complexity of QTI 1.x as are now being made with regard to QTI 2.x:

> There is no doubt that the QTI specification is highly complex, with some remaining apparent inconsistencies and ambiguity making it difficult to implement. [158, p. 324]

Finally, Sclater et al. also discussed the unattractiveness of interoperability for commercial vendors:

> While it is helpful to be able to claim that your product is interoperable it is not necessarily to your advantage as a vendor for it to be so. As well as adding to your system development costs your clients may ultimately decide to move to another system and use your interoperability feature to take their content with them. [158, p. 325]

With the emergence of strong open-source competition (especially from Moodle), the market situation has changed compared to 2002 (see the discussion of universities migrating away from commercial platforms on the facing page), but the implementation situation of QTI is effectively unchanged. Sclater's views may thus be interpreted as resignation in the face of the slow progress being made:

50. That is, Blackboard and WebCT, which are now both owned by Blackboard Inc.; WebCT is now marketed under the name of Blackboard Learning System – Vista Enterprise License.

51. Systems evaluated were Questionmark Perception, WebCT, Blackboard, Canvas Arena, and CETIS Rendering Tool.

> If there's nothing to interoperate with and if the content is effectively "future-proofed" by being exportable into XML this begs the question: does it matter at all whether Moodle properly adopts the IMS QTI specification? [157, p. 73]

We agree with Gorissen's analysis, but we do not think that the effort necessary for a reference implementation would be justifiable, since the problems are rooted in the design of QTI. Consequently, we think that a fundamentally different approach would be necessary to create an interchange format that meets the requirements outlined in section 4.5.2.

We also agree with Sclater's analysis in that institutions may not see see a pressing need for interoperability, whether because they do not see a need to exchange tests with others, or whether they do not see themselves moving to a different platform. However, we think that these believes are based on wishful thinking rather than on reality. The fact that, as Sclater mentions, more and more institutions are moving from commercial e-learning platforms to open-source platforms shows that even university-level strategic decisions may be revised, making interoperability suddenly critical. Here are some current examples of universities migrating their e-learning platforms from Blackboard or WebCT to Moodle [see also 175]:[52]

- Louisiana State University started migration from Blackboard to Moodle in 2008. The University decided the switch to Moodle was necessary because of financial reasons. [121, 171]

- Idaho State University (ISU) also decided to adopt Moodle to replace WebCT. The background given for this decision shows that a migration to another platform may not be solely a decision of the university:

  > In the fall semester of 2005, ISU was informed by WebCT that our current platform – WebCT CE 4.0 – would no longer be supported after July of 2007. The new version (WebCT CE 6.0) was identified as requiring a significant shift in personnel and equipment support and resources. Soon after the version 4.0 "end of life" and support deadline was announced, Blackboard Corporation, which makes a competing LMS product, announced its acquisition of WebCT. This merger was finalized in April of 2006 and created additional concerns for product directions, pricing, and support requirements. [74, p. 10]

- In the beginning of 2007, Brandeis University started a project aiming to replace WebCT—which had been in use at the university since 1997—with Moodle. After a three-year transition to WebCT Vista from the earlier "Campus Edition" version, WebCT proved to be unstable, a problem that was aggravated by a lack of support after the purchase of WebCT by Blackboard Inc. [65]

52. There are, of course, other commercial and open-source platforms besides the ones mentioned here; the exmples were chosen to illustrate Sclater's claims.

- In February 2008, the University of Kent started a project to migrate the university's of e-learning platform from WebCT to Moodle, since it was concluded that "future progress will require a move to a more modern and flexible platform."[53]
- In March 2008, the University of Puget Sound decided to adopt Moodle to replace Blackboard, which had been in use since 2003. The university's report [102] cites "gradual and growing faculty dissatisfaction with Blackboard, its outdated and poor feature set, its general 'buginess,' [*sic*] poor security and lack of integration with official course enrollment data, and low level of support from the vendor" as reasons for the decision to move to the new system.

The above examples show that migration to a new system is often not a deliberate decision, but forced by a variety of circumstances which are hard to foresee. Nevertheless, when running a mission-critical system—such as an e-learning platform of a university—one should have contingency plans; being able to move content from one system to another should be an important part of such a plan.

There is, however, an issue Sclater fails to mention: Interoperability is an important concern for instructors. When instructors move from one institution to another, they need to be able to continue using their educational material, including their tests.

Thus, for the university as operator of an e-learning platform, interoperability may be seen just as an insurance against vendor lock-in. Instructors, on the other hand, have a daily need for interoperability. We therefore think that there is a very real need for interoperability of questions and tests.

The current, unsatisfactory state shows, in our opinion, that those who need interoperability are unable to make themselves heard, *not* that there is no need for it. One factor may also be that the majority of instructors are "just users" and lack the technical knowledge to express their needs.

To summarize: QTI 2.0 can be considered a typical case of both "design by committee" and of "second-system effect" [23]: It was clearly not developed on the basis of actual usage or clear requirements, but it is rather a hodgepodge of every imaginable feature. As we have already suggested in [137], we think that a more promising approach would be if the users (i.e., the "educational community") got together and developed a more practical interchange format on the basis of their experience and their actual needs.

Quick Edit as Interchange Format?

The S-expression-based Quick Edit format developed for ECQuiz (see section 4.5.1) was primarily intended as an experimental format for authoring, not for interchange. Therefore there is currently no mechanism for packaging a test description with additional media, such as images. However, as the format is easy to generate, it has been used as

53. http://www.kent.ac.uk/elearning/moodle/announcement.html (accessed 2008-10-17)

a target format for conversion from other formats.[54] When tests contain no or only few images or other media, packaging is not necessary.

This use of the Quick Edit format indicates that it may also be suitable as a basis for a more comprehensive interchange format for tests. Using the base-64 encoding syntax and "display hints" defined in [147] (or a similar mechanism) it would be possible to include binary data, e.g., images, inside the test description. This would allow to encode a complete test, including supporting media, into a single file. Alternatively, a packaging mechanism could be devised that makes it possible to store additional files alongside the test description and to reference these files from the test description. This mechanism could be based on IMS Content Packaging, or a different, perhaps more lightweight, approach could be taken.

In its current form, the Quick Edit format is not yet suitable for use as a format for interchanging tests between delivery platforms and authoring systems. With carefully designed extensions, however, it could, form the basis of such an interchange format and represent an alternative to QTI.

4.6 Summary

In this chapter, we have first described Bloom's Taxonomy and two newer educational taxonomies; these taxonomies provide frameworks for categorizing learning objectives and specifically for the design of corresponding assessment activities. We have then given a general overview of e-assessment and its historical development. The main part of this chapter described the assessment-related eduComponents products in detail; for each product we have also outlined the educational objectives for the assessment of which they can be employed.

In an excursus, we have examined selected issues of test authoring and interchange and presented the experience in this area made during the work on ECQuiz.

The next chapter describes the actual use of the eduComponents products and the experiences made in their use.

54. For example, Moshe Rappaport (University of Hawaii) has used a Microsoft Word macro to convert Respondus Standard Format items to Quick Edit format for import into ECQuiz (personal communication, see `http://www.hawaii.edu/geog_mr/ecquiz-converter.html` (accessed 2008-10-17)).

5 Practical Use, Experience, and Evaluation

The eduComponents have been in use in the Knowledge-Based Systems and Document Processing Research Group (WDOK) at Otto von Guericke University Magdeburg since 2004. ECQuiz (initially called LlsMultipleChoice) was the first component to be put into service in the fall of 2004. The other eduComponents products were introduced successively. During winter semester 2006/2007, the complete eduComponents-based learning environment was used by over 200 students at WDOK.

From winter semester 2005/2006 until summer semester 2007 we actively gathered feedback from our students by conducting surveys: At the end of each semester students were asked to fill out a questionnaire on their experience with the learning environment (section 5.3). The questionnaire from summer semester 2007 is reproduced in appendix B.

The items of the questionnaires focused on the comparison of e-assessment—as supported by the eduComponents—with traditional methods of assessment and on the students' behavior in response to the new assessment methods. We have also gathered feedback from instructors (section 5.4) and non-WDOK users (instructors and system administrators, section 5.5).

Before discussing the evaluation results, we first describe the technical infrastructure and the use of the eduComponents in the courses taught at WDOK.

5.1 Infrastructure

During the period of time discussed here (winter semester 2005/2006 through summer semester 2007), Plone with the eduComponents ran on a Sun Fire V440 server with four 1.28 GHz UltraSPARC IIIi with 16 GB RAM and the Solaris 9 operating system.

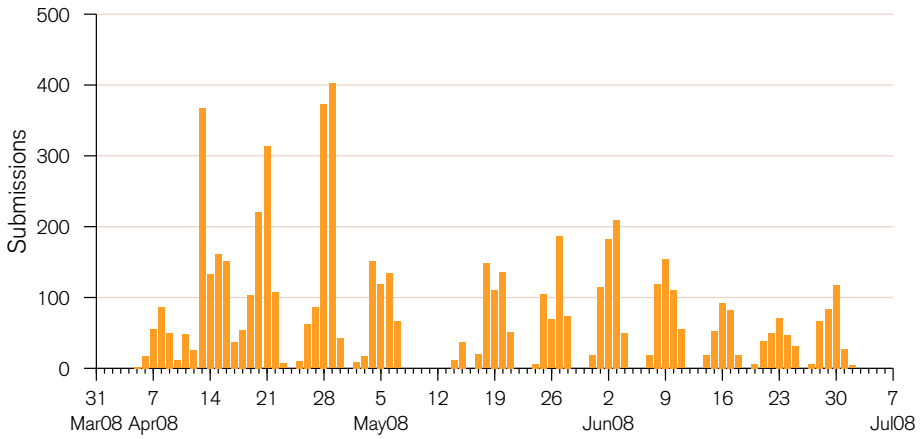


Figure 5.1: Submissions per day during summer semester 2008. The labeled ticks (April 7, 14, 21, etc.) are Mondays.

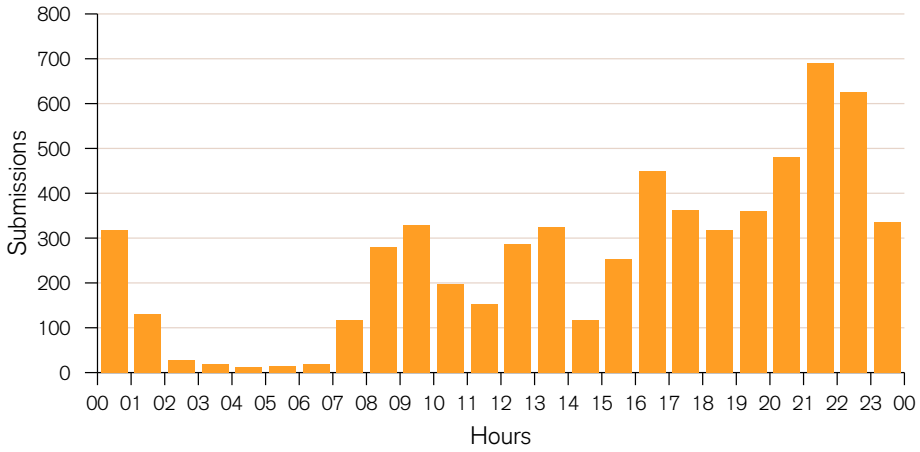


Figure 5.2: Distribution of submissions over the hours of the day during summer semester 2008.

ECSpooler and its backends ran on a separate system, a Fujitsu-Siemens SCENIC W620 workstation with a 3 GHz Intel Pentium 4 dual-core processor and 1 GB RAM, running the NetBSD 3.x operating system.

From a system load perspective, the use of the e-learning platform is characterized by peaks of submission activity, which depend on the deadlines of the courses. It is a well-known phenomenon that students tend to submit their solutions only shortly before the deadline, so submission numbers show a typical pattern, starting with low submission numbers and increasing toward the deadline.

Figure 5.1 shows the number of submissions per day during summer semester 2008 and illustrates this phenomenon. During that semester, 4 courses were offered, which took place on Monday evenings, on Tuesday mornings and afternoons, and on Wednesday mornings. The diagram shows that most submissions occured on Fridays (for the course on Monday), on Mondays (for the course on Monday and the courses on Tuesday), and on Tuesday (for the course on Wednesday). Submissions for the new assignments only started on Thursdays, with the numbers gradually ramping up.

Submission activity is not distributed evenly over the day, either. Figure 5.2 shows the distribution of submissions over the hours of the day during summer semester 2008. It can be seen that the numbers of submission increased towards the night, with a peak between 2100 and 2200 hours. The period between 2100 and 0000 hours accounts for over a third of all submissions, with the late afternoon and early

evening (1600–2000) accounting for another 24%. Thus, about 60% of the submissions were made between 1600 and 0000 hours.

These usage patterns directly translate to corresponding system load patterns: Periods of relatively low system load are followed by short periods, or "bursts," of very high load, when large numbers of users are simultaneously creating many new objects.

These patterns differ from normal CMS usage, which is mostly characterized by read-only accesses and only relatively infrequent creation of new objects.

The hardware and software setup was able to cope with the load, however with the larger numbers of students during winter semester 2006/2007 the system noticeably slowed down during periods of peak activity, as the single-server setup was not designed for this number of users. This problem is of course common to all types of applications and can be mitigated by appropriately scaling the infrastructure to the number of users. In the case of Plone this means, for example, using ZEO and load balancing to distribute the load over multiple servers. Work toward a higher-performance infrastructure was initiated at WDOK during summer semester 2007.

5.2 Usage

For readers unfamiliar with German universities, we will briefly describe the general organization of courses before the Bologna Reform, as the four semesters during which the evaluation took place were still organized in the traditional way with the *Diplom* as the first degree. While the details vary from university to university and from subject to subject, the basic structures are usually similar.

In German universities, many courses—undergraduate courses in particular—can be considered to consist of two parts: One part is a traditional *lecture* (*Vorlesung*), usually given by a professor. During lectures, students are mostly passively listening, and no credits are awarded for attending the lecture.

Accompanying the lecture are *exercise courses* (*Übungen*), usually taught by teaching assistants. Exercise courses aim to put the theoretical knowledge presented in the lecture into practice. Furthermore, credits for the course are usually gained through the exercise course; the requirements for credits may be the completion of a certain amount of coursework (possibly including an in-class presentation), a final exam, or a combination thereof. In WDOK courses, students had to work on weekly assignments (usually consisting of about 5 problems) and present their solution for a certain number (typically 2 or 3) of assignments in class. Class time is typically used for discussion of student solutions and review, and if necessary clarification, of concepts presented in the lecture.

Thus, at German universities, exercise courses are a cornerstone of teaching and learning and among the most interactive types of courses. What is more, most assessment takes place in exercise courses; exercise

courses are therefore most directly affected by the introduction of e-learning.

The traditional learning environment in computer science exercise course (and those of related subjects) is based on paper and blackboard. Students usually brought their hand-written notes to the exercise courses and presented their solutions at the board, with the teaching assistant commenting and the other students taking notes. This mode of writing and copying was time-consuming and error-prone. This especially applies to programming assignments: First, most non-trivial programs do not fit on a blackboard, second, the correctness of most non-trivial programs is hard to assess by just looking at them. Given the time constraints of exercise courses, only a limited number of solutions could be presented and discussed.

At Otto von Guericke University—but at other universities as well—a system called "voting" was used for gaining credits: At the beginning of the classroom session a list of the assignments due for this session was circulated and students marked the assignments they had—purportedly—worked on and would thus be willing to present in class. Since not every student could be asked to actually present his or her solution, this effectively allowed students to get points for assignments they had not actually completed.

With the introduction of ECAssignmentBox, the procedures in exercise courses were changed as follows: Like before, students received weekly assignments, but now using ECAssignmentBox instead of paper (or PDF) exercise sheets. Instead of voting, students were now required to submit their solutions via ECAssignmentBox some time *before* the session.

Teaching assistents now brought a notebook computer to class and used a data projector to display both the assignments and select student submissions. For each assignment a student was asked to present his or her solution; the corresponding submission was displayed using the data projector, so that students did not need to copy their solution to the blackboard. This resulted in a number of significant improvements:

- Passing the voting form around and having the students indicate the assignments they had worked on had taken a considerable amount of time, since teaching assistants would have to wait for the form to return before they could begin the session. Now the session could be started on time.
- Before, the teaching assistants had no idea which assignments had caused problems for the students. Likewise, they also did not know which assignments had been easy. It was thus possible that too much time was spent on the discussion of easy items and not enough time on difficult items. Teaching assistants also did not know about the solution strategies chosen by students, thus they had no idea of typical and atypical (e.g., especially elegant) solutions. Since the student to present his or her solution was effectively chosen at random, neither good nor bad solutions could be discussed on purpose. Since it was now possible to quickly review all submissions before the session, teaching assistants could get an overview and see, for example, how many

students had had problems with a particular assignment, and whether there were any frequently made mistakes. Now, teaching assistants could specifically ask for (pedagogically) interesting submissions to be presented.

- Copying solutions to the blackboard used to consume a good part of the class time, limiting the opportunity for discussion. There had usually been no time available to discuss more than one solution, so students had only rarely asked for the instructor's comments on their alternative solution. Since now presentation of the *full* solution was instant, more than one submission could be presented. After students realized this, they grabbed the opportunity to get feedback for their solutions.

- The paper-based system had allowed students to get points for assignments they had not actually completed, whereas they now had to submit written solutions for the assignments electronically before the classroom session. The old system had thus effectively not only encouraged fraud, but it had also discouraged achievement, as many good solutions would never even been seen by instructors. In fact we received positive comments from students on the fact that the system was now more "achievement-oriented."[1]

- Previously, student submissions had effectively been ephemeral, since they were only presented orally or because the graded submission was returned to the student. Now, submissions were archived electronically and were thus available for consultation, comparison, and various types of analyses previously impossible.

A further innovation applied to programming courses and courses with a significant amount of programming: For programming assignments, students were now required to submit working programs through ECAutoAssessmentBox. For these courses the above comments apply as well, but the effects are even more far-reaching: Instead of a rough sketch on the blackboard, students now had to submit not necessarily correct, but at least *running* programs.

Summarizing the most important changes, the demands for students' solutions became much more explicit and rigorous with respect to correctness, quality, and clarity. On the other hand, students could now benefit from access to a larger number of alternative solutions and to typical error cases.

While it would theoretically be possible to enforce these requirements in a paper-based system, it would result in an unmanageable workload for the teaching assistents.

1. For example, in winter semester 2006/2007, a student put this comment on their questionnaire: "Besonders gut fand ich, dass das System bis auf einige Ausnahmen leistungsorientierter ist. Man kann Aufgaben nicht mehr wie in der EAD-Übung auf gut Glück votieren, man muss zumindest schon eine korrekte Bearbeitung eingesendet haben." ("I especially appreciated that the system is, with only a few exceptions, more achievement-oriented. You can no longer try your luck as you could in the EAD lab course and just vote for any assignment, at the minimum you must have submitted a correct solution.")

The eduComponents-based e-learning environment, however, relies on electronic documents stored and managed in a CMS. Thus, assignments and student submissions can quickly be retrieved, analyzed, annotated, and presented. For both programming and essay-like assignments, reviewing larger numbers of student submissions is now possible because the submissions are collected at a central, network-accessible location. Instructors can easily browse and inspect them before the exercise course, so that specific problems observed in the submissions can be addressed in the course. Since all submissions are now available online in the exercise course, solutions can easily be presented and compared; faulty solutions to programming assignments can be corrected and immediately tested. The time spent formerly to write sketchy solutions onto the blackboard is now free for discussion.

5.3 Student Questionnaire Results

The effectiveness of an e-learning platform is hard to assess objectively. However, to get a general impression of student attitudes towards our eduComponents-based e-learning environment we asked our students to fill out questionnaires in winter semester 2005/2006, summer semester 2006, winter semester 2006/2007, and summer semester 2007. The questionnaire in winter semester 2005/2006 was an online pilot survey; the experience from the pilot survey was used to refine the initial design of the questionnaire.[2]

In the following three semesters, we used a paper-based questionnaire, which was filled out during the last class session of the semester. During these three semesters the questionnaire remained very much stable, both with respect to the ordering and the wording of questions, apart from some minor changes, e.g., to find about specific issues observed during the semester. The questionnaire for winter semester 2006/2007 is reproduced in appendix B on page 181.

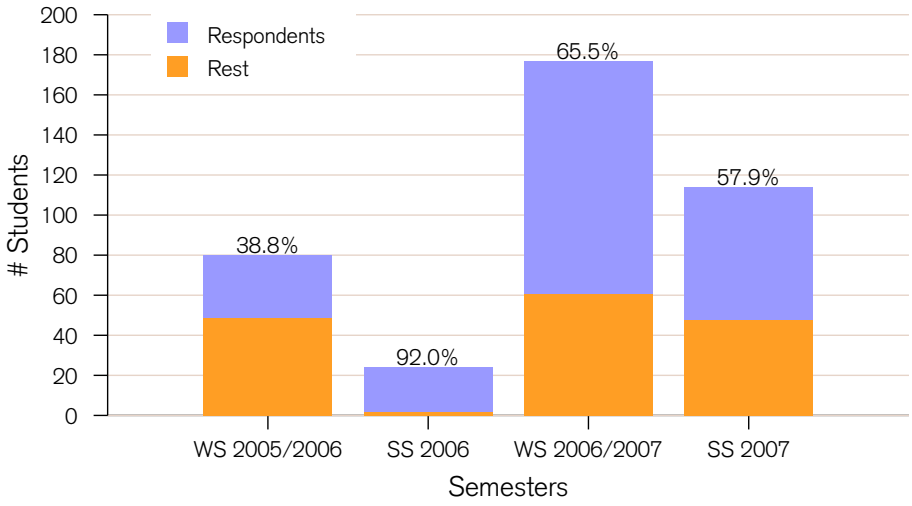


Figure 5.3: Effective numbers ("active" students only) of students using eduComponents at WDOK.

We chose to use a paper questionnaire to be able to better control the response rate. Figure 5.3 shows the response rate in comparison to the total number of "active" students, i.e., students who regularly partici-

2. The questionnaires were jointly developed by the author and Mario Amelung.

pated and fulfilled or approached the requirements for credits.[3] The response rate is still below 100% because of drop-outs and absences in the class session during which the questionnaire was distributed[4].

The students participating in WDOK courses, and thus the respondents, were mostly students of computer science or closely related subjects. The only exception was winter semester 2006/2007, which includes 23.9% (predominantly female) students of media education[5], also raising the proportion of female students to 31.3% for this semester; the average proportion of female respondents for the other two semesters was 10.6%. The total number of respondents over the three semesters was 204. Since e-learning platforms were not in general use at Otto von Guericke University at that time, respondents were generally not familiar with other platforms.

In this dissertation we restrict ourselves to the discussion of selected results relating to three key areas:

1. **Usability:** Are the eduComponents easy to use?
2. **Organization:** Does the eduComponents-based learning environment help students organize their learning?
3. **Effects on learning process:** Does the use of the e-learning platform influence students' learning processes?

For each of these areas we have selected the questions most relevant to the respective areas from the questionnaire. Figures 5.4 to 5.8 on pages 152–154 graphically show the results over three semesters.

There are two types of questions in this set: Rating questions and yes-no questions. Rating questions are on a scale from 1 to 6, where 1 = "I fully agree" and 6 = "I strongly disagree"; all of these questions have the same polarity, so that the ratings can be interpreted like German school grades (1 = "excellent," 6 = "insufficient").

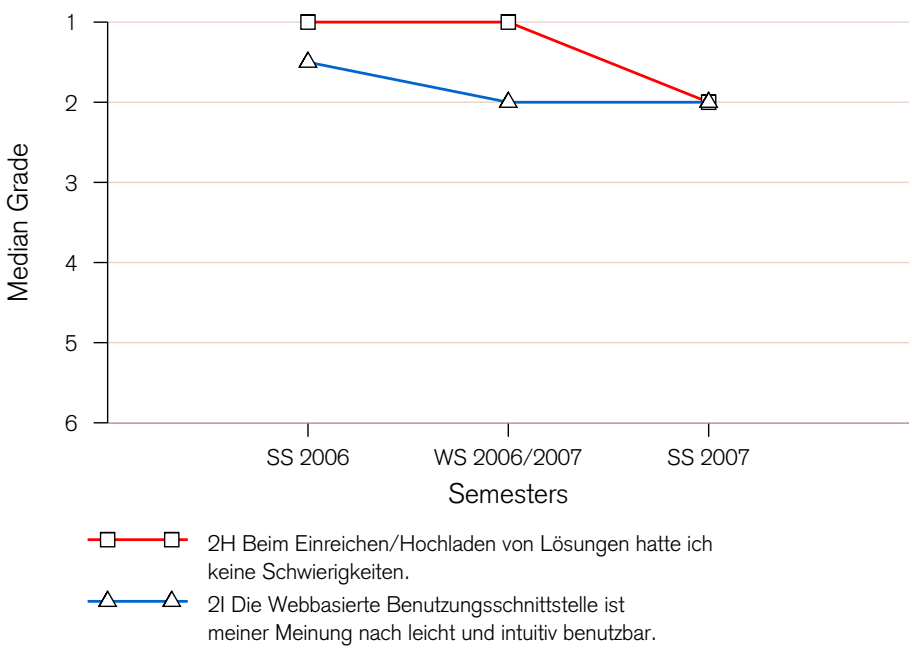
The graphs for the rating questions are all based on *median grades* since the median best represents the most typical rating and is more robust in the presence of outlier values than the mean.

5.3.1 Usability

The questions related to the usability of the eduComponents are 2H "I had no problems uploading and submitting my solutions" and 2I "The Web-based user interface is intuitive and easy to use."[6] As figure 5.4 on the next page shows, over the three semesters students very much agreed with these statements.

3. Since we are only interested in students who actually used the eduComponents, it is more accurate to use the number of active students rather than the total number of enrolled students.
4. Absentee rates are usually higher at the end of the semester: Some students who already have collected enough points choose not to come; others have to prepare for exams; in other cases there are conflicts with other courses.
5. The official name of the degree program is *Medienbildung: visuelle Kultur und Kommunikation* ("Media education: Visual culture and communication").
6. The codes refer to the question numbers on the winter semester 2006/2007 questionnaire reproduced in appendix B on page 181.

Figure 5.4: Answers to usability-related questions 2H "I had no problems uploading and submitting my solutions" and 2I "The Web-based user interface is intuitive and easy to use" over three semesters (1 = "fully agree," 6 = "strongly disagree").



It cannot be said whether there actually was a decline in agreement as the diagram may seem to suggest. If there was, in fact, a decline, it may have been due to the slowdown of the platform caused by inadequate computing resources.

5.3.2 Organization

The questions related to the topic of whether the e-learning platform helped students organize their learning are 2C "The online availability of assignments and submissions in a central location is very helpful" and 2E "The system offers a good overview of the number and the state of my submissions." Figure 5.5 on the facing page again shows consistent agreement with these statements.

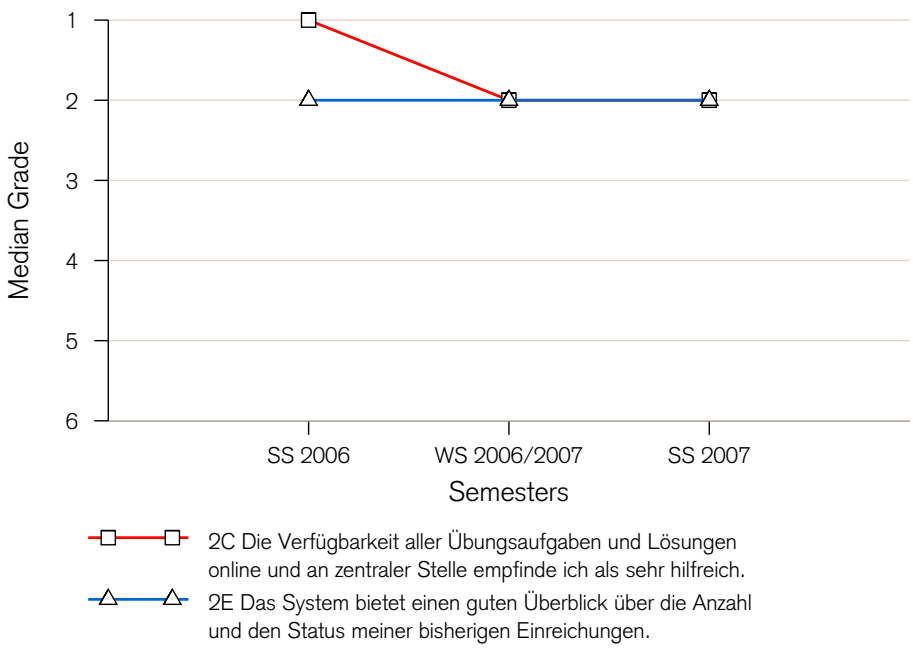
5.3.3 Influence on Learning Processes

The questions related to influences on learning processes are 3B "I work more diligently on my assignments than in courses without electronic submissions" (figure 5.6 on the facing page) and 3C "My solutions are more elaborate than before" (figure 5.7 on page 154).

These statements could only be answered with *yes* or *no*. Our rationale was that an answer on a scale would be hard to interpret, both for respondents and for evaluators, unless the meaning of the scale were precisely defined: For example, what would a 3 mean in response to question 3C? That the respondent's solutions were *somewhat* more elaborate,[7] *sometimes* more elaborate,[8] or that the respondent was *not quite sure* whether they were more elaborate?[9] We feel that a yes-no question, on the other hand, gives a better overall impression of stu-

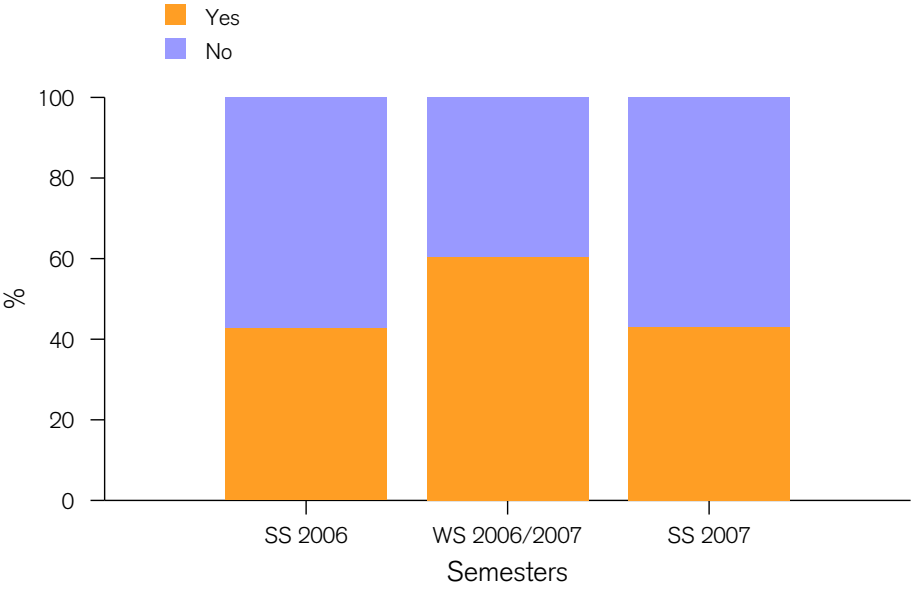
7. Interpreting the question as "How much more elaborate are your solutions?"
8. Interpreting the question as "How often are your solutions more elaborate?"
9. Interpreting the question as "How sure are you that your solutions are more elaborate?"

Figure 5.5: Answers to organization-related questions 2C "The online availability of assignments and submissions in a central location is very helpful" and 2E "The system offers a good overview of the number and the state of my submissions" over three semesters (1 = "fully agree," 6 = "strongly disagree").



dents' feelings toward their questions because it requires respondents to make a clear decision.

Figure 5.6: Answers to question 3B "I work more diligently on my assignments than in courses without electronic submissions" over three semesters.



Considering figures 5.6 and 5.7 on the next page, we can see that, while in most semesters only a minority of respondents agreed with these statements, there was still a considerable percentage (around 40%) who did. Interestingly, in winter semester 2006/2007 over 60% of the students said they worked more diligently on electronic submissions. It is also noteworthy that the answers to these two questions very much differed from one course to another; however, no explanation can be given for this observation, apart from the speculation that the effect depends on course content and assignment types.

Two further questions also relate to the influence of the eduComponents on learning processes: 4B "The automatic evaluation motivates

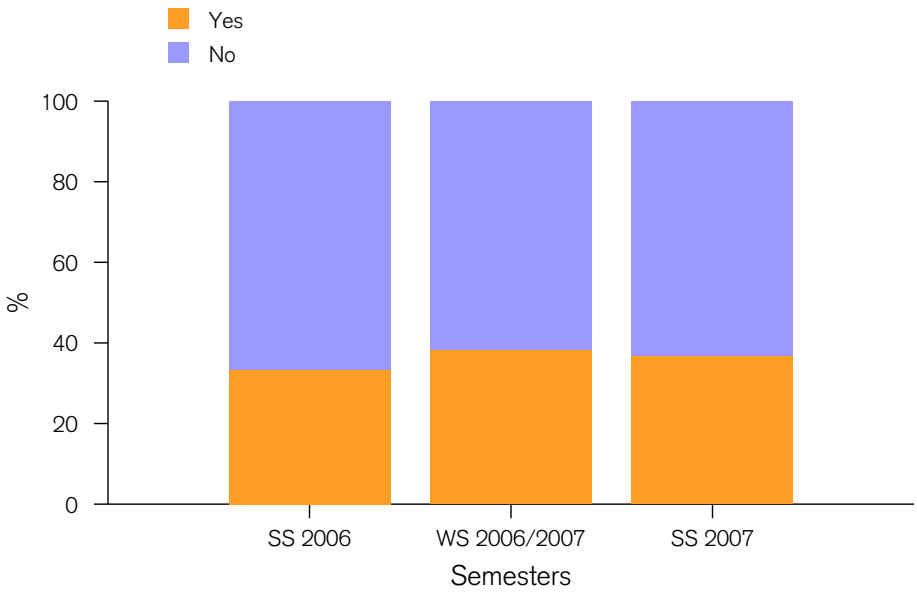


Figure 5.7: Answers to question 3C "My solutions are more elaborate than before" over three semesters.

me to write running and correct programs" and 4C "The automatic feedback helps me to find errors in my programs." These two questions cover the effects of the automatic evaluation of programming assignments as implemented by ECAssignmentBox and ECSpooler.

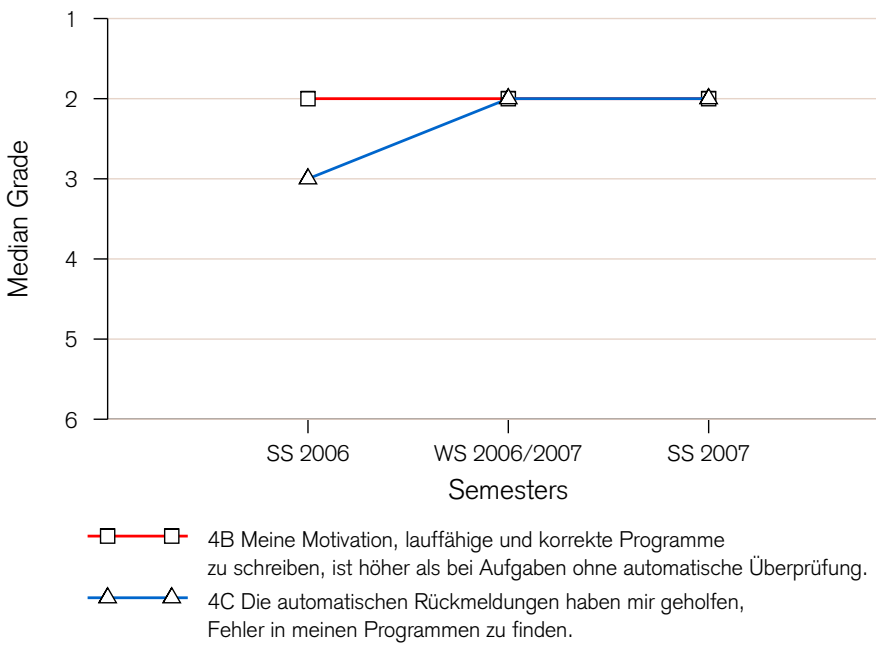


Figure 5.8: Answers to questions relating to the effects of automatic feedback on the learning process: 4B "The automatic evaluation motivates me to write running and correct programs" and 4C "The automatic feedback helps me to find errors in my programs" (1 = "fully agree," 6 = "strongly disagree").

Amelung [2] gives an extensive description of these components and a more detailed evaluation of their use. However, as they are part of the eduComponents, the answers to the above two questions are briefly discussed.

As can be seen in figure 5.8, students gave consistently good ratings for the automatic evaluation of programs over the three semesters. The agreement with statement 4C improved from the first deployment of the system in summer semester 2006, which may be due to the

instructors becoming more familiar with automatic testing and thus being able to create better test cases.

5.4 Comments from Instructors

Feedback from instructors at the WDOK Research Group was collected through the method of *informal interviews* [114, p. 56].

Instructors commented that the administration and review of student submissions was much easier with the eduComponents, and that they had a better overview since tests, assignments, and submissions are collected centrally.

Instructors also reported that electronic submissions helped them in the preparation of face-to-face sessions and in the early detection of problems. They repeatedly stressed that electronic submissions saved time previously required for copying solutions to the blackboard during face-to-face sessions.

Overall, the acceptance of the eduComponents-based e-learning environment by instructors was very high at WDOK.

5.5 User Questionnaire Results

Besides the attitudes of our students, another important aspect in evaluating the eduComponents is their use by others outside the WDOK Research Group and the Otto von Guericke University.

After the release of the eduComponents as open-source software and corresponding announcements on relevant mailing lists, the eduComponents were quickly picked up by other users. We therefore set up a mailing list, `educomponents@uni-magdeburg.de`, to offer off-site eduComponents users a forum for questions and discussions.[10] The archives of the list are publicly accessible.[11] At the time of this writing, the mailing list counts about 80 members from a wide variety of institutions all over the world.

Examples include the Cranionline online course in traumatology of the Virtuelle Hochschule Bayern (vhb), where ECQuiz is used, Duborg-Skolen in Flensburg, the boarding school of the Danish minority in Germany, where ECQuiz, ECLecture, and ECAssignmentBox are used, the Faculty of Architecture at the Center for Environmental Planning and Technology, Ahmedabad, India (using ECLecture and ECAssignmentBox), the Chairs for Genome-Oriented Bioinformatics and Machine Learning and Data Mining in Bioinformatics at the Technische Universität München (Munich, Germany), the Institute of Mathematics at the Johannes Gutenberg University Mainz, Germany, and the British Royal School of Military Survey.

The flexibility of the component-based approach is reflected by the diversity of applications, which go beyond "typical" learning institutions

10. The term *users* in this section means instructors and system administrators using eduComponents products at their institution, not students.
11. `https://listserv.uni-magdeburg.de/mailman/listinfo/educomponents` (accessed 2008-10-17)

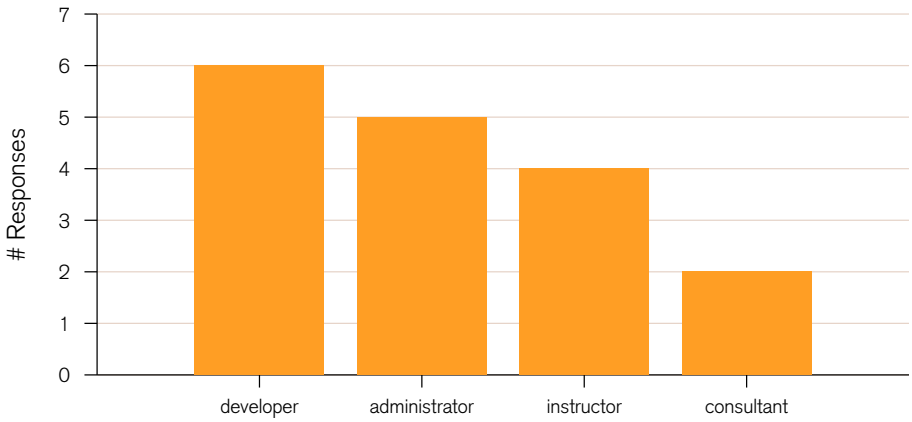


Figure 5.9: Roles of the eight respondents to the user questionnaire; multiple responses were possible.

and include VCScollaborate.org, an online collaborative platform for the voluntary and community sector in Great Britain, the Corporació Catalana de Ràdio i Televisió (the Catalonian broadcasting corporation) in Barcelona, Spain, and the Waterloo-Wellington Flight Centre (a flying school) in Waterloo, Canada.

The mailing list members have provided important feedback during the development of the eduComponents, and from their comments it can be deduced that they are in general very satisfied with the eduComponents products.

To gather somewhat more general feedback (i.e., not only related to one specific product or issue) on how the eduComponents are being used at other institutions we have posted a questionnaire to the mailing list. The questionnaire is reproduced in appendix C.

We have received eight responses, i.e., a response rate of 10%. The affiliations of the respondents appear to correspond to the mix described above: 4 respondents were affiliated with universities (in Germany, the U.K., and the U.S.), 2 with companies in Germany and the U.S., 1 with an EU development project in Egypt, and 1 with an advanced vocational training institute in Italy.

The questionnaire first asked respondents to identify their role or roles with respect to the eduComponents; respondents answered as follows (see figure 5.9):

4 Instructor

2 E-learning consultant

5 System administrator

6 Software developer

Most of the respondents thus have multiple roles, and none of the eight respondents saw themselves as instructors only. We think that this is due to the fact that the mailing list is primarily read by users who at least install the software themselves; we know from previous contributions that some members are also able to modify or extend components. A dual role of instructor and part-time system administrator or software developer is also typical for assistants in scientific and technical departments at universities.

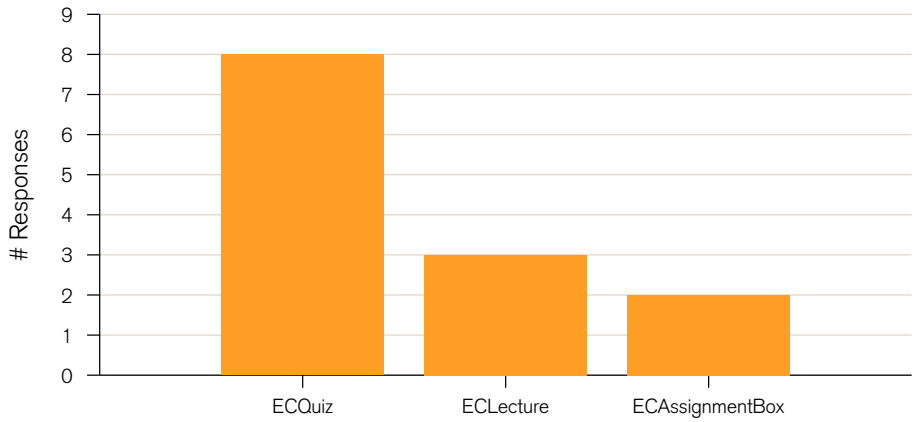


Figure 5.10: Products used by the respondents to the user questionnaire; multiple responses were possible.

Regarding the answers to the questionnaire, the high level of technical knowledge of the respondents is likely to give more insights into the reasons for using eduComponents products.

The respondents indicated use of the following products (see figure 5.10):

3 ECLecture

8 ECQuiz

2 ECAssignmentBox

0 ECAutoAssessmentBox and ECSpooler

0 ECReviewBox

ECQuiz is clearly the most popular product, but it is also the one that has been available for the longest time. The components for automatic testing (ECAutoAssessmentBox, ECSpooler, and ECReviewBox) were only recently released and are inherently more specific in use and more complex to set up. To our knowledge there are currently only two other users of these components, namely the University of Rostock and the Ludwig-Maximilian University Munich; we have received separate feedback from these users, which is discussed in detail by Amelung [2].

The primary purpose of this questionnaire was to investigate *why* people decide to use the eduComponents and *how* they use them—to find out whether the component-based, document-oriented approach of the eduComponents actually provides a viable alternative to conventional e-learning platforms.

The questionnaire therefore explicitly asked for "specific reasons or requirements" for using the eduComponents. All of the eight respondents named the eduComponents's integration in Plone in response to this question. Both the possiblity to seamlessly combine eduComponents products with existing Web content managed in Plone and to use the same infrastructure for e-learning and non-e-learning content were mentioned. All of the respondents also indicated that they are using Plone as a content management system; one respondent also said his company was using Plone as a knowledge management system.

Most of the respondents indicated that they were not using other e-learning software besides the eduComponents, which shows that users can and do select exactly the modules they need, and that these modules indeed satisfy their e-learning needs.

One respondent, affiliated with a university that uses ILIAS as a centralized e-learning platform, answered:

> I have been using ILIAS but I don't like the idea of having to maintain two systems – ILIAS for the e-learning and Plone for the rest of our Web pages.

The respondent thus confirms our view that the separation of learning content and "other" content, as required by conventional e-learning platforms, is a problem, and that the integration of e-learning components with CMS-managed content is preferrable.

In the final question, "What do you like best about the eduComponents? Do you have any suggestions or comments?," respondents also emphasized the advantage of the integration of the eduComponents with Plone. Some comments were:

> I like the integration into Plone a lot. The code base is very readable. It is not what one would consider bloat-ware.
>
> Ich mag die Einfachheit der Komponenten und deren Einbindung in Plone.[12]
>
> I like the fact that they are customisable and sit comfortably within the Plone workflow set-up

Respondents proposed a few new features and wished for Plone 3.x compatiblity. One respondent, who had previously submitted patches to ECLecture, suggested that the eduComponents code should be made available through a public Subversion[13] repository. A few specific outstanding bugs were mentioned, but there was no large-scale criticism. The problems mentioned by the respondents are being addressed in the current development at WDOK or will be addressed in future revisions.

5.6 Summary and Discussion

This chapter has outlined the usage of the eduComponents at the WDOK Research Group and reported on evaluation results based on responses to questionnaires by students at Otto von Guericke University Magdeburg and on informal interviews with WDOK instructors. We have also reported on feedback from eduComponents users outside Magdeburg.

The students' responses were collected over three semesters. We have shown selected results related to the three areas of usability, organizational support, and influence on learning processes.

The results vary only slightly over the three semesters and are consistently positive. The usability of the eduComponents-based e-learning

12. "I like the simplicity of the components and their integration with Plone."
13. Subversion is a revision control and source code management system.

platform was highly rated. Students valued the reporting and statistics features and found it helpful that their assignments were stored centrally and easily accessible. About 40% of the respondents also reported that they worked more diligently on their assignments and that they were more elaborate than before.

Instructors at WDOK reported significant time savings and numerous improvements in their courses through the use of eduComponents products.

The eduComponents are open-source software and can be freely downloaded and used. The use of eduComponents products by numerous institutions world-wide is in itself a good indicator that our approach provides a viable alternative to typical learning management systems.

All respondents to a questionnaire posted on the eduComponents mailing list stressed that the integration of e-learning with general content management, as offered by the eduComponents, was an important factor for them. In general, we can conclude that the respondents are very satisfied both with the eduComponents approach and the functionality of the individual components.

The feedback from students and instructors at Otto von Guericke University and from eduComponents users at other institutions provides strong evidence that the eduComponents approach works and represents a practical alternative to conventional e-learning platforms. Most importantly, the responses validate the theses we have put forward in section 1.3. The feedback constitutes further proof for Thesis 1, namely that a large portion of the functionality commonly found in e-learning platforms can be abstracted from its domain-specific usage and thus provided by a content management system extended with assessment functionality.

Experience and feedback from instructors and administrators from WDOK and other institutions in particular support items (2) and (3) of Thesis 2: The flexibility of the component-based architecture of an e-learning environment based on the eduComponents is highly rated.

6 Outlook

In the preceding chapters we have presented the concepts, the design, and the implementation of the eduComponents, as well as an evaluation of the eduComponents in practical use.

In this chapter we will outline further applications of the eduComponentsand potential directions for future development.

The essence of the eduComponents approach are the *concepts* for document-oriented e-learning components described in this thesis (in particular in chapters 3 and 4). In fact, the concrete implementation as Plone products may be considered as merely a proof of concept. Unlike typical prototypes, however, the eduComponents products are in production use at a number of sites. This vouches for the quality of the implementation, but, more importantly, it is further evidence that our approach is viable and solves actual problems.

Thus, while the Plone implementation is an important part of our work and our argumentation, the eduComponents concepts could, in principle, also be realized on other platforms, as long as they fulfill the technical and conceptual requirements.

6.1 eduComponents and the Plone roadmap

eduComponents development began on Plone version 2.1. When Plone 2.5 was released in 2006, all products were upgraded to also work with this version, since it offered notable benefits with regard to speed and stability.

In August 2007, Plone 3.0 was released, the first release of the new 3.x series, introducing a large number of new features; the release notes for Plone 3.0 summarized the long list[1] as follows:

> Aims to make Plone more efficient to work with. This release adds versioning, locking, inline editing and validation, link integrity, intranet/extranet workflows, wiki support, OpenID sup-

1. `http://plone.org/products/plone/features/3.0/` (accessed 2008-10-17)

port, and full-text indexing of Word/PDF. (`http://plone.org/products/plone/releases/3.0` (accessed 2008-10-17))

Plone 3.x also introduced new low-level methods and mechanisms, some of which require changes in products, e.g., in product initialization code. Furthermore, a number of methods was deprecated and should be replaced by the corresponding new methods.

At the time of this writing, work has begun to make the eduComponents products compatible with Plone 3.x, as there have been many requests for it on the eduComponents mailing list[2]. Once this has been achieved, the next logical step is to revise the eduComponents code to harness the new facilities offered by Plone 3.x.

Our component-based approach allows the eduComponents to benefit from improvements of the underlying platform. For example, like any Plone product, the eduComponents products benefit from performance improvements. The same applies to user interface optimizations: Through the use of *AJAX* (asynchronous JavaScript and XML) technologies, content creation in Plone 3.x is much quicker than before, without requiring any effort from product authors.

Plone 3.x also introduced "pluggable" support for markup syntaxes, i.e., the support is realized in the form of plug-in modules, which can be added or removed by system administrators.[3] Plone currently ships with modules for plain text, HTML, Structured Text, ReStructured Text, Markdown, and Textile formats, and for some specialized formats. By creating plug-in modules, developers can provide support for further formats. Again, products can take advantage of this new capability with no or very low effort from product developers. For the eduComponents this is a welcome addition, as it can be used to further lower the demands on students and for creators of e-learning materials, as it allows instructors and students to use their preferred style of markup; on the other hand, if desired, this capability can also be used to enforce the use of a specific markup syntax.

Other new facilities may allow improvements and further refinements of the existing eduComponents products; for example, by taking advantage of the built-in versioning support of Plone 3.x, it may be possible to manage multiple submissions to an assignment box as versions of a single submission, which would enable the use of the built-in version management tools of Plone. Obviously, versioning would also be interesting for assignment authors.

However, while there are many improvements in Plone 3.x—and further planned enhancements—that the eduComponents could benefit from in one way or another, we think that these are not crucial for the eduComponents *concept*. Our experience has successfully shown that the document-oriented, component-based approach of the eduComponents is feasible, and that a CMS, and Plone in particular, constitutes a highly suitable platform for e-learning applications. Thus, the question is now how further developments in Plone could *substantially* enhance our approach beyond gradual improvements as described above.

2. See section 5.5 for more information about the eduComponents mailing list
3. Previously, support for a fixed number of markup formats was built into Plone.

We see two main areas where such enhancements are possible:

1. As soon as a non-negligible amount of learning content has been produced, it is desirable to make this content easily reusable.

   While Plone, as a CMS, provides all the basic functionality to serve as *learning object repository* or *item bank*, the work described in this dissertation was focused on other aspects of the eduComponents.

   It would therefore be interesting to examine and to evaluate the current usage of Plone as a learning object repository and to determine how to best make use of Plone's facilities for this specific application.

2. Since all submissions made by students are available electronically and since Plone, as a content management system, naturally offers all functionality required for managing content, a further interesting direction to explore is the use of Plone and the eduComponents for creating and managing *e-portfolios*.

Both points are discussed in more detail below.

6.2 Plone as a Learning Object Repository

6.2.1 What are Learning Objects?

Packaging and distribution of learning content is moving towards sub-course units, so-called *learning objects*. The goal is to enable more efficient reuse, re-aggregation, and personalization of learning content, as well as (commercial) creation, distribution, and mass customization.

As with most e-learning terms, there are various definitions for what exactly constitutes a *learning object*. As a formal standard, IEEE 1484.12.1 "Learning Object Metadata" (LOM) [69] suggests itself as a source for an authoritative definition:

> Learning Objects are defined here as any entity, digital or non-digital, which can be used, re-used or referenced during technology supported learning. [69]

However, as Wiley [184] points out, this would make both Joan of Arc and a banner ad over a Web page of an online course legitimate learning objects. Since the IEEE definition is clearly too broad, Wiley instead proposes the definition of a learning object as "any digital resource that can be reused to support learning." This definition has the advantage of being much more specific and practicable, and it remains compatible with the IEEE definition since it defines a proper subset of learning objects as defined in IEEE 1484.12.1.

Polsani [138], however, criticizes Wiley's definition as "a simple case of uncritical nomenclature without conceptual clarification," on the grounds that "classifying every digital asset as a LO nullifies the basic features of modularity, separation of content and context, and reusability borrowed from object-oriented programming." Polsani instead proposes the following definition:

> A Learning Object is an independent and self-standing unit of learning content that is predisposed to reuse in multiple instructional contexts. [138]

Even though it has become clear that it is difficult to determine and specify what exactly a learning object is or should be, at this point, an intuitive understanding of the term *learning object* is sufficient, and Polsani's definition should be adequate for the following discussion.

Since the idea of learning objects is that they can be reused and recombined as needed by instructors or students, or automatically by some software, it is necessary that learning objects are accompanied by metadata that enables users (whether humans or software) to find learning objects corresponding to their needs[4].

The above-mentioned IEEE LOM standard's purpose is to define a "common conceptual data schema" [69, p. 5] to ensure that learning objects can be exchanged and that different systems handling learning objects are interoperable. It is debatable whether the standard actually achieves this goal: On one hand, it defines a large number of metadata elements, which makes annotation of actual learning objects challenging. On the other hand, many of these elements and their value spaces are only vaguely defined or subjective; Downes [48], p. 6 criticizes: "Much of the metadata in IEEE-LOM could be classified as subjective metadata." One example is "semantic density," which is described as:

> The degree of conciseness of a learning object. The semantic density of a learning object may be estimated in terms of its size, span, or—in the case of self-timed resources such as audio or video—duration.
>
> The semantic density of a learning object is independent of its difficulty. It is best illustrated with examples of expositive material, although it can be used with active resources as well. [69, p. 26]

The "semantic density" element can take the values "very low," "low," "medium," "high," or "very high." The standard itself concedes that "this scale is meaningful within the context of a community of practice," but even this caveat disregards the inherent subjectivity of such a classification; Downes notes:

> In any case where such a subjective assessment is called for (and there are many more), we are automatically presented with the possibility of differing descriptions for any given resource. One observer may describe a learning object (or a movie) as 'too complicated for average viewers', while another may say it is 'challenging but accessible'. [48, p. 6]

Thus, given the difficulty of determining the "semantic density" of a real-life learning object, we think that this scale will rarely be meaningful at all, at least for interchange beyond very small communities.

Nevertheless, IEEE LOM provides a common frame of reference and is currently the metadata standard referenced in some way by practically

4. In the case of software agents these are, of course, the user's needs as modeled in the system

all projects and implementations related to learning objects. A popular alternative is Dublin Core [76], which is a simpler and more generic metadata set, but there exists a proposed mapping between IEEE LOM and Dublin Core [172]. Seth [161] provides an overview of different learning object metadata schemas.

The third entity in the learning object "ecosystem" is the *learning object repository* (LOR). A learning object repository can be considered a kind of digital library for storing, managing, and sharing learning objects. Just like a traditional library, a learning object repository primarily relies on metadata to enable access to its holdings, so it should consistently and reliably apply a metadata standard such as IEEE LOM. Unlike traditional libraries, learning object repositories do not necessarily need to store the actual learning objects, and some repositories may choose to store only metadata and references to learning objects if these are accessible via the Internet.

The term *learning object repository* is often used to refer both to repository software (e.g., Fedora [97] or dLCMS [155]) and to specific repositories, their holdings, and their metadata schemas (e.g., ARIADNE[5] [49] or Merlot[6] [151]). As repository software was often originally developed for a specific project, there are close connections between the software and its architecture and the repository in the latter sense. For an overview, Neven and Duval [123] offer a comparative study of 10 learning object repositories. Seth [161] discusses a number of repositories, specifically with respect to their metadata schemas.

While it is likely that many researchers would agree with the observation of Sicilia et al. that "[t]he concept of reusable learning object has become the central notion of a new approach to education and organizational learning that emphasizes reusability as the key characteristic," [164, p. 466] it is not equally likely that everybody agrees with their conclusion that this approach will result in "a general improvement in quality and efficiency when designing learning experiences."

Before coming back to the question posed on page 163 (i.e., can Plone in conjunction with the eduComponents be considered or used as a learning object repository?), we therefore insert a brief critical discussion of the concept of learning objects.

6.2.2 Excursus: Learning Objects—A Critical Discussion

A basic issue with the concept of learning objects is the underlying assumption (rarely made explicit in technical publications on learning objects) that (1) learning content can, indeed, be decomposed into small pieces (or created as small pieces), and that (2) such pieces—coming from different sources—can in fact be reassembled to form meaningful learning content. Or, in other words, *granularity* and *composition*.

Another issue with respect to granularity is that the effectiveness of some learning object (whatever its size) is highly dependent on its context. For example, a diagram may be well suited to enhance a textual

5. *ARIADNE* stands for "Alliance of Remote Instructional Authoring and Distribution Networks for Europe."
6. *Merlot* stands for "Multimedia Educational Resource for Learning and Online Teaching."

explanation but be useless when used on its own. However, context is not as easy to share and reuse as data. The question how learning objects can be combined is thus also a question of instructional design [see 184].

Authoring educational content in the form of learning objects, i.e., in small chunks and relying only on as little context as possible is a challenging task. Since a learning object "must embody explicit planning for learning, intentional instructional design" to have an educational impact [66], it is debatable whether the production of learning objects fulfilling these requirements is possible without teams of specialists working according to detailed plans.

At universities, learning material is typically created by instructors for their own courses, which take place in a very specific, known context. Material is often created only shortly before it is needed, thus adapting to the requirements of the students and the course on the basis of experiences made and feedback received. Content is reused by instructors, but only rarely shared; it is constantly adapted to new requirements, evolving from one semester to the next. In this scenario, which is typical for universities, it is hard to imagine instructors creating reusable learning objects and annotating them with extensive metadata, since it would require a large amount of extra work with no immediate benefit.

Of course, this traditional mode of production is criticized, not only by publishers who would like to sell more content to universities [e.g. 135], but also by academics. For example, Polsani argues that, "as the nature and functional requirements of knowledge are ever-changing in the knowledge economy, no single academic or subject expert can generate a total knowledge adequate to the tasks." Instead, he advocates that "knowledge experts should develop only 'events' of knowledge that can combine with other 'events' to develop into a 'program' on demand" [138], using the term *event* instead of *learning object*.

In any case, besides issues of attitude, there are still numerous open questions with respect to learning objects; for example, along with a number of other issues, Dodani [46] points out that a "good indicator of our inexperience with learning objects is the myth that a good way of chunking a learning object is by time." In other words, it is still unclear what the optimum granularity of learning objects is and how it is to be determined. Furthermore, due to problems ranging from interoperability problems to structural and legal issues, there are not yet many learning object repositories that actually contain readily reusable learning objects; Verhaart notes:

> If the vast body of current theoretically based literature is synthesized, it could be concluded that there are repositories containing packaged learning object materials that could be used in a personal or corporate Learning Content Management Systems. This is true for targeted LCMS's such as Blackboard or WebCT. However, it appears that the Repositories have taken a different direction and have become records of meta-data stored in a database. [178, p. 462]

In fact, Merlot, for example, contains only links to educational content of typically very large granularity. After a survey of objects from several repositories, Harvey [66] summarizes that "relatively few can strictly be defined as LOs [learning objects] at all, being of a basic CO [content object] type and not useable on a stand-alone basis to bring about intentional learning." He also notes that the quality of objects "varies greatly, and in some cases calls into question the professionalism associated with the learning materials of which they are a part." He concludes:

> The current review indicates that the effective development of LOs requires the clear definition of an instructional process addressing the unique characteristics of LO technologies, within the structured process stressed by ID [instructional design] principles. If such principles are not heeded, learning repositories will gain a reputation for amateurish content, rather than credibility as worthwhile educational resources. [66]

To summarize: As Dodani notes, "learning objects are still at their infancy," and it is currently not yet clear whether their theoretical potential can be achieved, especially in today's university environments.

6.2.3 Learning Object Repositories vs. CMSs

Now, to come back to the question whether Plone in conjunction with the eduComponents can be considered or used as a learning object repository, we would argue as follows: Content management systems are designed for managing reusable content objects (e.g., articles or news items), for annotating them with metadata, and for searching and retrieving objects on the basis of their content or their metadata (see section 2.4). These features are very similar, if not identical, to those of a learning object repository. We also contend that whether a content object is a learning object solely depends on its purpose. We can therefore conclude that a CMS can indeed serve as a learning object repository. In fact, Connexions[7] (Rice University) and eduCommons[8] (Utah State University) are learning object repositorie (if one wants to use this term) built on Plone.

However, the functionality of a CMS—such as Plone—goes beyond that of a repository, as a CMS integrates facilities for authoring and publishing content objects. Since a CMS separates content from presentation, objects can be presented differently depending on their context; for example, only the headline of a news item may appear on the frontpage, or an article may be made available in HTML for display, in PDF for download, and as plain text for an e-mail newsletter. Thus, a CMS is not just a catalog of objects but it includes the facilities for processing (including rendering) the objects as well.

The question is then, what could be the advantage of dedicated learning object repositories. An article entitled *I already have a virtual learning environment, why do I need a learning object repository?* in a newsletter of Intrallect, a U.K. vendor of learning object repository software, gives two reasons:

7. `http://cnx.org/` (accessed 2008-10-24)
8. `http://plone.org/products/educommons` (accessed 2008-10-24)

> So the first reason for storing learning objects separately from the VLE is that they are likely to be used in many other contexts where a VLE is not used.
>
> The second reason for separating the functions of a learning object repository from a VLE is that it reduces proprietary tie-in to that VLE. This not only makes it possible to change from one VLE to another without worrying about losing the embedded content, but it offers the flexibility to work with multiple VLEs at the same time [. . .] [75]

While these reasons are meant to promote the sales of LOR software, they make obvious that learning object repositories are effectively a way to circumvent shortcomings of conventional e-learning platforms. Since, by design, the eduComponents do not have these shortcomings, there is no need to store content separately.

To summarize: In this section we have discussed the use of Plone as a learning object repository. After defining the terms *learning object* and *learning object repository*, we have critically analyzed the ideas behind these terms and concluded that they are not yet fully developed. The concept of the learning object repository is, at first glance, convincing. On closer inspection, however, it rather seems to be a workaround for limitations of conventional e-learning platforms.

The eduComponents approach does not have these limitations, since it is *document-oriented* instead of course-oriented, meaning that learning content exists independent of its use in a concrete course. The CMS features of Plone enable more flexible management, processing, and reuse of content than either LORs or conventional e-learning platforms. What is more, the document-oriented approach of the eduComponents offers significant new possibilities: Unlike conventional learning object repositories (and e-learning platforms), the eduComponents treat instructor-created content and student submissions uniformly. This means that the repository contains both assignments and the corresponding student submissions, which in turn opens new opportunities for evaluation and reuse.[9]

Thus, as we have noted above, Plone with the eduComponents can be regarded and used as a learning object repository—but its actual functionality goes significantly beyond that of conventional LORs.

6.3 Using Plone and the eduComponents for E-Portfolios

The uniform treatment and storage of all types of (learning) content also makes Plone with the eduComponents suited as the technical basis for a "personal repository," or *e-portfolio*.

E-portfolios are the electronic version of *portfolios*, which have, especially in the liberal arts, long been used for assessment and demonstration of achievement; the JISC e-assessment glossary [24] defines a portfolio as "a representative collection of a candidate's work, which is used to demonstrate or exemplify either that a range of criteria has been met, or to showcase the very best that a candidate is capable of." A

9. There is currently work underway in the WDOK research group to further explore this potential [see 129].

portfolio can contain a variety of artefacts, such as "samples of writing, both finished and unfinished; photographs; videos; research projects; observations and evaluations of supervisors, mentors and peers; and reflective thinking about all of these." [25, p. 2] Butler emphasizes reflective learning, i.e., "it is the reflections on the pieces of evidence, the reasons they were chosen and what the portfolio creator learned from them, that are the key aspect to a portfolio." [25, p. 2]

Mason et al. [112] identify three main uses (and corresponding types) of portfolios: For development, for presentation, and for assessment purposes.[10] A portfolio may be related to a particular course or be constructed over a longer period of time; long-term portfolios are often associated with the concept of *life-long learning*.

Conceptually, e-portfolios are based on the same ideas as traditional paper-based portfolios, but they can take "advantage of the capabilities of ICT, notably allowing learners to store digital artefacts and streamlining the process of review and moderation for learners, tutors, moderators and verifiers." [24] Similarly, Mason et al. [112] see the advantage of e-portfolios in " the number and quality of services that can be provided to individuals and the community, the portability and adaptability of the output and the potential to create central repositories."

Especially interesting in our context is that Mason et al. likens e-portfolios to learning object repositories:

> E-portfolios are not dissimilar, in the technical sense, to a collection of learning objects. That is, they assemble discrete pieces of electronically available material that can be manipulated, stored and re-versioned to suit different audiences. Just as learning objects can be stored in a learning object repository, students add new work to their e-portfolios over the period of their study and select the most appropriate items from this repository to present at the end of their course, or to an employer at the end of their degree. Adhering to standards allows these repositories (e-portfolios) to be re-usable over time, within different systems, conveniently accessible to different audiences as required.
>
> [112, p. 719]

With Plone and the eduComponents, this usage is natural, since all content is treated uniformly and can be reused and presented in various ways. Unlike conventional e-learning platforms, Plone and the eduComponents are also suitable for publishing content. Mason et al. conclude that "e-portfolios can be a fitting assessment model for courses designed in learning objects. E-portfolios consist of discrete pieces of work and this mirrors the structure of learning objects, particularly those which are activity based" [112, p. 726]. In our view, the eduComponents approach supports this method in an ideal way.

10. Barrett [10] and Butler [25] discuss the various purposes for which portfolios may be used in detail.

6.4 Summary

In this chapter we have explored two potential directions for further development and use of the eduComponents. One direction is that of learning object repositories, the other that of e-portfolios.

Following a discussion of learning objects and learning object repositories, we have concluded that an e-learning environment based on Plone and the eduComponents is in fact functionally and architecturally superior to conventional learning object repositories. However, specialized support for managing e-learning content could be added to provide domain-specific functions for end-users, e.g., Plone could be made aware of learning-specific metadata items (e.g., difficulty) and user interfaces could offer corresponding values for selection.

We have also briefly explored the idea of using Plone and the eduComponents for e-portfolios. Given the definitions and uses of e-portfolios, we have concluded that an eduComponents-based e-learning environment ideally supports the use of e-portfolios as an educational device due to the content management facilities provided by Plone. As in the case of learning object repositories, the necessary functionality is already available, but a further eduComponents product could offer end-users domain-specific functionality, e.g., a special user interface for the creation or review of e-portfolios.

7 Summary and Conclusion

Based on an analysis of the current state of the art in e-learning platforms and of the problems associated with these software systems, we have put forward two theses in the beginning of this dissertation:

Thesis 1: A large portion of the functionality commonly found in e-learning platforms can be abstracted from its domain-specific usage and can be shown to be common content management functionality.

Thesis 2: The architecture of systems that can be used for e-learning has to be modular. Modularity is required on three levels: (1) On the level of implementation, allowing programmers to create new components, (2) on the level of administration, allowing system administrators to specify which components to install and to add third-party modules, and (3) on the level of end users, i.e., allowing instructors to select and combine components to obtain the functionality they need to realize the pedagogical goals via e-learning.

We have argued in chapter 2 that, by using abstraction, the functionality of typical e-learning platforms can be shown to consist of (1) basic content management functions and (2) assessment functions.

Since content management systems (CMS) offer more general and more robust functions for managing content, this means that e-learning platforms can—or even should—be built on robust and proven content management systems. Only assessment functions are actually specific to e-learning and need to be added to a CMS; this requires the architecture of the CMS to be modular.

As a proof of concept, we have designed and implemented the eduComponents, a component-based e-learning system architecture, realized as software components extending a general-purpose content management system with facilities for course management and assessment. The design and implementation of the eduComponents have been described in chapters 3 and 4. The individual development of the

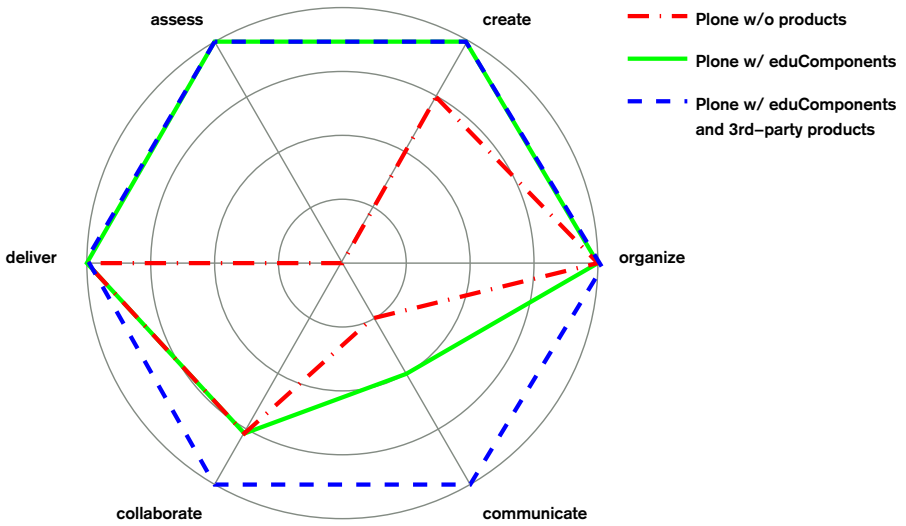


Figure 7.1: Radar plots (introduced in section 2.2.1) can be used to visualize the level of support systems provide for the six activities required for e-learning platforms. This plot displays the approximate coverage of activities offered by Plone, by Plone with the eduComponents, and by Plone combined with the eduComponents and additional third-party products. The radial coordinates should be read as "no support" at the center and "extensive support" at the outermost circle.

various eduComponents products shows the advantage of modularity on the implementation level (item (1) of Thesis 2).

The eduComponents differ from typical research prototypes in that they are used productively for teaching and learning at Otto von Guericke University Magdeburg and at other institutions world-wide. Chapter 5 provides an evaluation of the eduComponents in actual use. The evaluation is based on students' answers to questionnaires, collected over a period of three semesters, results of informal interviews with instructors, and responses of eduComponents users (instructors, developers, administrators) at other institutions.

The evaluation results especially support items (2) and (3) of Thesis 2 and demonstrate that the eduComponents approach is not only conceptually valid but that the eduComponents can also successfully be used to realize flexible e-learning platforms and allowing the integration of e-learning and Web content management on a *single* platform.

In section 2.2.3 we have defined e-learning platforms as systems providing support for six activities—creation, organization, delivery, communication, collaboration, and assessment—in an educational context. We have also introduced the visualization of the actual level of support of an e-learning platform for theses activities in the form of a radar plot (see figure 2.2 on page 42).

Figure 7.1 visualizes the support for the six activities offered by Plone, by Plone with the eduComponents, and by Plone combined with the eduComponents and additional third-party products (e.g., for discussion forums, wikis, or chats).

Figure 7.1 thus illustrates that by (1) abstracting the content management functionality found in e-learning platforms from its domain-specific usage and mapping it to the more generic functionality of a content management system, and by (2) utilizing a component-based architecture to selectively add functionality specific to e-learning, namely assessment functionality, it is possible to achieve better support for all of the six activities than commonly offered by conventional e-learning platforms. The eduComponents approach has the additional benefit that functionality can be installed selectively, so that the e-learning environment can be tailored to the actual needs.

A further aspect—and another advantage—of our component-based approach is that components from different sources can be combined.

7.1 Component Synergy

During development and use of the eduComponents we have realized that a component-based architecture offers advantages beyond the obvious ones, such as flexibility, modularity, and reusability, namely *synergy*.

Software components are functional units communicating through well-defined interfaces. This means that a component can communicate with any other component that implements the same interfaces. No internal knowledge about other components is needed, neither when the component is written, nor at run time. This is obviously an essential property to enable interoperability and interchangeability.

We use the term *synergy* to describe the effect that the addition of a component not only adds the functionality of the component but that new functionality *emerges* by the interaction or "cooperation" of the components, resulting in combined functionality greater than the sum of their separate functionalities.[1] Synergy is highly desirable because it offers users new functionality through an existing user interface and without creating dependencies between components.

One example of synergy in the eduComponents is the "Lecture view" of the standard Plone folder. When ECLecture is installed, folders acquire a specialized view option: Instead of the standard listing of the folder contents, a folder containing ECLecture objects can act as a course directory, as shown in figure 7.2 on the next page. All of the information shown comes directly from the ECLecture objects contained in the folder; this means that course data (title, instructor, location, etc.) only has to be entered once and it is ensured that the course directory is always current. Thus, from the synergy of ECLecture and Folder emerges the new functionality of a course directory.

Another example is the "Today's Lectures" portlet. Portlets are small content containers that can be placed in a specific area of a Plone site, as shown in figure 7.3 on the next page. The location and content of the portlet area can be configured by the site administrator. When ECLecture is installed, a new portlet, "Today's Lectures" (see figure 7.3) becomes available; as the name says, the portlet lists the lectures taking place on the current day (i.e., ECLecture objects with the appropriate properties), serving both as a reminder of the time and the place and as a quick link to the lecture home page (the ECLecture object) and thus all information and materials related to the lecture. Here, new functionality emerges from the cooperation of ECLecture and the portlets mechanism.

1. In analogy to the definition of *synergy* in the *New Oxford American Dictionary* (2nd ed.) as "the interaction or cooperation of two or more organizations, substances, or other agents to produce a combined effect greater than the sum of their separate effects." The term *component synergy* can be found in computer science literature, albeit surprisingly rarely; for example, Kaminka and Tambe [88] speak of "synergistic interactions, which allow each component to function beyond its original capabilities because of the presence of other components."

Figure 7.2: The "Lecture view" is an example of component synergy between the standard Plone folder and ECLecture. It provides the functionality of a course directory by automatically extracting the relevant information from contained ECLecture objects and presenting it in a convenient format.

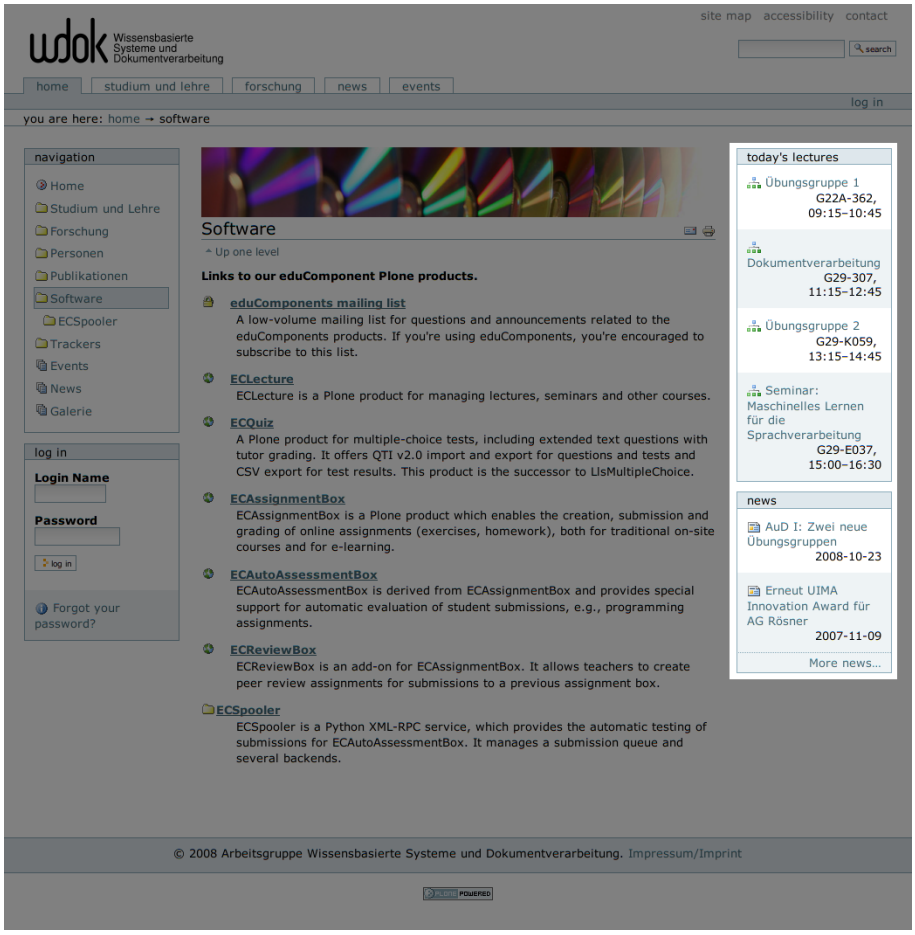
Wintersemester 2006/2007

Up one level

Lehrangebot im Wintersemester 2006/2007

title	course type	instructors	time	recurrence	location
Dialogue Structure in Human-Machine Interaction	Seminar	Dietmar Rösner, Milan Gnjatović	Tuesday, 15:00–17:00	weekly from 2006-10-10 until 2007-01-23	G29-E037
Natural Language Systems I	Lecture	Dietmar Rösner	Friday, 15:00–16:30	weekly from 2006-10-20 until 2007-01-26	G29-037
Programmierkonzepte und Modellierung	Vorlesung	Dietmar Rösner	Tuesday, 11:15–12:45	weekly from 2006-10-10 until 2007-01-23	G29-307
Advanced Concepts in Document Processing	Seminar	Michael Piotrowski, Dietmar Rösner	Tuesday, 17:00–18:30	weekly from 2006-10-24 until 2007-01-26	G29-E037
Lehr- und Lernsysteme	Vorlesung	Dietmar Rösner	Thursday, 13:15–14:45	weekly from 2006-10-12 until 2007-01-26	G29-307
Dokumentverarbeitung	Vorlesung	Dietmar Rösner	Friday, 11:15–12:45	weekly from 2006-10-13 until 2007-01-26	G29-307

Figure 7.3: Portlets are small content containers that can be placed in a specific area of a Web portal. This screenshot shows the portlet area of a Plone site (highlighted), containing two portlets ("Today's Lectures" and "News").



However, e-learning environment based on the eduComponents could be further improved by exploiting component synergy to a greater extent. Interfaces play an important role in enabling synergy: By implementing as many of the applicable interfaces as possible, components allow more components to take advantage of their functionality.

A good example is the Plone Event interface. This interface is implemented by the standard Event content type (for describing events that take place at a certain time and place); it is used by the Plone calendar and the "Upcoming Events" portlet, but also by add-on products, such as CalendarX[2].

A lecture is, of course, a prime example of an event. We had therefore considered adding an Event interface to ECLecture, which would make ECLecture objects behave like events (and be displayed in the calendar, for example). Unfortunately, recurring events, e.g., events that occur weekly or every first Monday of a month for a certain period of time, are not yet supported by Plone. For seminars, lectures, and other courses, this is rather the rule than the exception; the lack of support for recurring events would have made the behavior of events represented by ECLecture objects unintuitive and misleading.

Support for recurring events is currently on the Plone roadmap[3]. Once realized, ECLecture could implement the Event interface with little effort. Built-in support for recurring events would enable even closer integration of eduComponents-provided e-learning functionality with the Plone infrastructure and open up interesting possibilities for course management through synergy with other add-on products supporting the Event interface.

Yet to explore are further opportunities for synergies for ECLecture through interfaces defined by third parties, for example with the Booking[4] product for booking of rooms and resources or the Faculty/Staff Directory[5] product for more advanced management of information about instructors.

7.2 Concluding Remarks

A common assumption in e-learning research and practice is that e-learning requires special e-learning platforms, such as Moodle or Blackboard. It is thus assumed that these platforms provide unique services not provided by other software systems.

In this dissertation we have questioned these assumptions and we have argued that the functionality of conventional e-learning platforms can be characterized by the simple formula "Web + assessment," i.e., e-learning platforms implement basic content management and other common Web functionality (such as forums, chats, wikis, etc.) and

2. `http://plone.org/products/calendarx` (accessed 2008-10-17)
3. Roadmap item #158. The Plone roadmap (`http://plone.org/products/plone/roadmap` (accessed 2009-01-09)) is an overview of proposed features and the releases for which they are projected. The roadmap is maintained by the Plone development team.
4. `http://plone.org/products/booking` (accessed 2008-10-17)
5. `http://plone.org/products/faculty-staff-directory` (accessed 2008-10-17)

functionality for assessment (such as quizzes). On closer inspection it is obvious that only assessment functionality is actually specific to e-learning—technically, a forum is a forum, no matter what it is used for. Furthermore, the content management and communication functionality in e-learning platforms is typically restricted and often inferior when compared with the more general implementations available in Web content management systems.

The eduComponents approach presented in this dissertation represents a novel design and has resulted in a usable implementation, which has been in actual use at Otto von Guericke University and other institutions since several semesters. The experience with the eduComponents gives practical proof of the theses we have put forward in this dissertation and of the feasibility of the eduComponents approach.

The research done for this dissertation has also resulted in practical definitions for *e-learning* and *e-learning platform*, terms which are notoriously ill-defined. Based on these definitions, we have proposed an innovative approach for assessing, visualizing, and comparing the functionality of e-learning platforms. Most attempts to evaluate and compare e-learning platforms are based on large numbers of criteria and individual features; however, since e-learning platforms differ widely in focus and functionality, they typically fail to produce clear results. The six activities characterizing e-learning platforms, which we have introduced in this dissertation, provide a convenient frame of reference for describing the functionality of e-learning platforms and related systems. The six activities can naturally be visualized as the six dimensions or "spokes" of a radar plot. Radar plots are an established technique for visualizing multi-faceted data but, to our knowledge, have not yet been used in the evaluation of e-learning platforms. Since each system is characterized by a specific shape in a radar plot, it allows to quickly ascertain and compare the functional focus and the strengths and weaknesses of different systems.

Some of the possible directions for further research and for further development of the eduComponents have been outlined in chapter 6 and in the preceding section. On a more general level, we hope that the results and insights presented in this dissertation may prompt a rethinking of the architecture and the aims of e-learning environments. On an even more abstract level, we would be delighted if our contributions to e-learning research could trigger a critical evaluation and reconsideration of e-learning and its technical implementation, eventually giving rise to completely new approaches and bringing us closer to realizing the true potential of e-learning.

A Glossary

Assignment In common usage, *assignment* is often used not only to denote "a task or piece of work assigned to someone as part of a job or course of study" (New Oxford American Dictionary, 2nd ed.), but also the work produced for the assignment (as in, "hand in your assignments").

To avoid misunderstandings, we use the term *assignment* only for the task and refer to the resulting work as ↗ *submission*.

Component Also *software component*. We follow Szyperski et al., who define the term as follows:

> A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.
>
> [173, p. 41]

Computer-assisted instruction (CAI) Also *computer-aided instruction*. ↗ ERIC descriptor: "Interactive instructional technique in which a computer is used to present instructional material, monitor learning, and select additional instructional material in accordance with individual learner needs." CAI can be considered both a predecessor and a specific use of ↗ *e-learning*.

Content management system (CMS) In this dissertation used as a synonym for *Web content management system*. A CMS is a system for managing content and associated metadata in a repository and for publishing this content. CMSs typically provide user and access management, asset management (including versioning), workflow management, search and retrieval, and a template-based publication facility, providing separation of content, structure, and presentation. For a more extensive discussion see section 2.4 on page 45.

Document For the ↗ *Plone* term, see ↗ *folder*.

E-assessment "The end-to-end electronic assessment processes where ICT is used for the presentation of assessment activity and the recording of responses. This includes the end-to-end assessment process from the perspective of learners, tutors, learning establishments, awarding bodies and regulators, and the general public." [24]

E-learning Unfortunately, there is no single definition for the term *e-learning*. We use it as a general term to describe all kinds of computer-mediated and computer-supported learning and teaching, regardless of whether it is used exclusively, as in distance learning situations, or additionally, as in face-to-face, instructor-led courses (*blended learning*). For a more extensive discussion see section 2.1 on page 25.

"The process of learning which is supported by the use of ICT (e.g. the Internet, network, standalone computer, interactive whiteboard or portable device). Also used loosely to describe the actual content delivered on-screen, and the more general use of ICT to contribute to learning processes." [24]

ERIC Education Resources Information Center `http://eric.ed.gov/` (accessed 2008-10-17) is an online database operated by the Computer Sciences Corporation (CSC), under a contract with the US Department of Education's Institute of Education Sciences providing access to education-related literature and resources. ERIC uses a controlled vocabulary of education-related terms called Descriptors that are maintained in the ERIC Thesaurus. The ERIC Thesaurus, browsable via `http://eric.ed.gov/thesaurus` (accessed 2008-10-17) is not only useful for searching the ERIC collection, but it also provides concise definitions of education-related terms; since many terms are used inconsistently and with different meanings throughout the literature, ERIC descriptors can be used as basic definitions to establish a frame of reference.

Folder Plone provides two basic classes of content objects: Documents and folders. The distinction is analogous two the distinction between directories and files in most file systems: Folders are containers that can contain documents and other folders, whereas a document can only contain "content," e.g., a text, an image, etc. Classes derived from the basic Plone folder class (ATFolder) are called *folder-like*. In Plone, the distinction is, however, not as sharp as in typical file systems: A folder-like object can also have content of its own.

Instructor We use *instructor* as a general term for a person supervising some educational activity, such as teaching a course, organizing assessment, etc. The actual term for such a person depends on the concrete circumstances and responsibilities may, e.g., be *professor*, *lecturer*, *tutor*, *teacher*, or *teaching assistent*.

Python A dynamic object-oriented programming language. Python runs on a wide variety of platforms, including UNIX systems, UNIX-like systems, such as NetBSD or Linux, and Microsoft Windows. Python is distributed under an open-source license. The

Python Software Foundation (PSF) holds and protects the intellectual property rights behind Python. The Python home page is `http://python.org/` (accessed 2008-10-17). ↗ *Zope* is mostly written in Python, as are Zope products (extension modules), including ↗ *Plone* and the eduComponents.

Plone A content management framework based on ↗ *Zope*. Plone is licensed under the GNU General Public License. The Plone home page is `http://plone.org/` (accessed 2008-10-17).

Portlet A small content container that can be placed in a specific area of a ↗ *Plone* site, especially for "live" information or information related to the main content that is being displayed. The term is not specific to Plone.

Product A Zope term for ↗ *component*.

Student In the context of the eduComponents, *students* is used as a general term for users who submit solutions to assignments, take tests, etc. in contrast to ↗ *instructors*, who create or supervise assignments and tests.

We generally prefer the term *student* over *learner* on the same grounds as Merrill et al.:

> Students are persons who submit themselves to the acquisition of specific knowledge and skill from instruction, learners are persons who derive meaning and change their behavior based on their experiences. All of us are learners, but only those who submit themselves to deliberate instructional situations are students. [115, p. 6]

Submission We use this as a general term for the work or solution submitted by a student in response to an ↗ *assignment*.

User The term *user* has many meanings in computer science: In the context of e-learning platforms, it can refer to a student, an instructor, a system administrator, a programmer, or an institution; furthermore, it can also refer to an entity managed by the e-learning platform, identified by a unique ID and associated with certain properties, such as permissions. To avoid confusion, we have tried in this dissertation to use the more specific terms (such as instructor, student, etc.) whenever possible; in the remaining occurrences, we have tried to make the meaning of the term *user* clear through the context in which it occurs.

Workflow The New Oxford American Dictionary, 2nd ed., defines *workflow* as "the sequence of industrial, administrative, or other processes through which a piece of work passes from initiation to completion." In an information technology context, a workflow usually a (business) process consisting of a sequence of steps, controlled by a program (a *workflow management system*) which models the workflow and assigns each task to a user responsible for this task. [See, e.g., 177] ↗ *Content management systems* usually provide support for the management of publishing workflows (including such steps as authoring, review, publication).

Zope An open-source Web application server for building content management systems and other Web applications. Zope is primarily written in the ↗ *Python* programming language. Zope was originally developed by Zope Corporation. Parts of Zope have been open-source since 1996; the whole application server is open source since 1998. The Zope home page is `http://zope.org/` (accessed 2008-10-17).

B Student Questionnaire

On the following two pages we reproduce the student questionnaire from summer semester 2007, which is discussed in chapter 5. The questionnaires for summer semester 2006 and winter semester 2006/2007 only vary slightly.

Markieren Sie so: ☐☒☐☐☐☐
Korrektur: ☐■☐☒☐☐

Verwenden Sie einen Kugelschreiber, rote Farbe unbedingt vermeiden!
Dieser Fragebogen wird maschinell erfasst. Bitte beachten Sie im Interesse einer optimalen Datenerfassung die links gegebenen Hinweise beim Ausfüllen.

Fragebogen zur E-Learning-Unterstützung in Lehrveranstaltungen

Mit diesem Fragebogen wollen wir auswerten, welche Erfahrungen Sie mit dem Einsatz von E-Learning in Lehrveranstaltungen der Arbeitsgruppe Wissensbasierte Systeme und Dokumentverarbeitung (Prof. Rösner) gemacht haben. Bitte nehmen Sie sich einen Moment Zeit und beantworten Sie folgende Fragen.

1. Zunächst möchten wir Sie bitten, einige allgemeine Angaben zu machen.

a.) Sie sind …? ☐ weiblich ☐ männlich

b.) Sie sind im …? ☐ Grundstudium ☐ Hauptstudium

c.) Welchen Studiengang belegen Sie? ☐ IF ☐ WIF ☐ CV ☐ DKE ☐ MVK ☐ anderer

d.) Für welche Lehrveranstaltung beantworten Sie diesen Fragebogen? ☐ IE ☐ NLS II ☐ PGP

e.) Welchen Abschluss der Lehrveranstaltung streben Sie an? ☐ Schein ☐ Prüfung ☐ keinen Abschluss

2. Folgende Fragen beziehen sich auf die elektronische Einreichung von Übungsaufgaben.

	trifft zu ... trifft nicht zu
a) Die elektronische Einreichung ist bequem.	☐ ☐ ☐ ☐ ☐ ☐
b) Elektronisch vorhandene Einreichungen sparen Zeit in den Übungen.	☐ ☐ ☐ ☐ ☐ ☐
c) Die Verfügbarkeit aller Übungsaufgaben und Lösungen online und an zentraler Stelle empfinde ich als sehr hilfreich.	☐ ☐ ☐ ☐ ☐ ☐
d) Die Geschwindigkeit des Systems ist ausreichend.	☐ ☐ ☐ ☐ ☐ ☐
e) Das System bietet einen guten Überblick über die Anzahl und den Status meiner bisherigen Einreichungen.	☐ ☐ ☐ ☐ ☐ ☐
f) Durch das System habe ich einen besseren Überblick über meinen Lernfortschritt.	☐ ☐ ☐ ☐ ☐ ☐
g) Die Online-Einreichung ist meiner Meinung nach ein sehr guter Ersatz für das bisherige Votiersystem.	☐ ☐ ☐ ☐ ☐ ☐
h) Beim Einreichen/Hochladen von Lösungen hatte ich keine Schwierigkeiten. Falls doch, welche Probleme hatten Sie?	☐ ☐ ☐ ☐ ☐ ☐

i) Die Web-basierte Benutzungsschnittstelle ist meiner Meinung nach leicht und intuitiv benutzbar. ☐ ☐ ☐ ☐ ☐ ☐

j) Ich würde es sehr begrüßen, wenn auch in anderen Lehrveranstaltungen solche Möglichkeiten angeboten würden. ☐ ☐ ☐ ☐ ☐ ☐

3. Im Folgenden interessieren uns die Auswirkungen der elektronischen Einreichung auf Ihre persönliche Arbeitsweise.

a.) Dass Lösungen einige Stunden vor dem Übungstermin eingereicht werden müssen, ist für mich kein Problem.
☐ Ja ☐ Nein

b) Ich löse die Übungsaufgaben gewissenhafter als in Übungsveranstaltungen ohne elektronische Einreichung.
☐ Ja ☐ Nein

c) Meine Lösungen sind ausführlicher als bisher.
☐ Ja ☐ Nein

d) Dass Lösungen schriftlich ausformuliert werden müssen, ist für mich kein Problem.
☐ Ja ☐ Nein

e) Die Übungsaufgaben bearbeite ich …
☐ fast immer allein ☐ meistens allein ☐ meistens mit anderen zusammen ☐ fast immer mit anderen zusammen

bitte wenden!

4. Sofern in der von Ihnen besuchten Lehrveranstaltung Übungsaufgaben mit automatischer Überprüfung von Programmieraufgaben angeboten wurden, beantworten Sie bitte auch die folgenden Fragen.

	trifft zu ... trifft nicht zu
a) Die automatische Überprüfung von Programmieraufgaben mit automatischer Rückmeldung ist insgesamt gesehen sehr hilfreich.	☐ ☐ ☐ ☐ ☐ ☐
b) Meine Motivation, lauffähige und korrekte Programme zu schreiben, ist höher als bei Aufgaben ohne automatische Überprüfung.	☐ ☐ ☐ ☐ ☐ ☐
c) Die automatischen Rückmeldungen haben mir geholfen, Fehler in meinen Programmen zu finden.	☐ ☐ ☐ ☐ ☐ ☐
d) Durch das System habe ich einen besseren Überblick über meinen Lernfortschritt.	☐ ☐ ☐ ☐ ☐ ☐
e) Die automatische Überprüfung hat mir geholfen, meine Programmierkenntnisse zu verbessern.	☐ ☐ ☐ ☐ ☐ ☐

5. Vergleichen Sie den Ablauf der Übung mit anderen von Ihnen besuchten Übungen ohne elektronische Einreichung. Was ist Ihnen aufgefallen?

a) Besonders gut fand ich...

b) Besonders schlecht fand ich...

Vielen Dank für Ihre Mitarbeit.

C User Questionnaire

The following message[1] was sent to the eduComponents mailing list[2] on June 12, 2008. The responses to the questionnaire are discussed in section 5.5.

Dear members of the eduComponents mailing list,

You may know that I am one of the two original creators of the eduComponents (together with Mario Amelung). What you probably don't know, is that the eduComponents are a topic of our research: They are, in a way, a proof of concept to demonstrate that component-based e-learning environments are a viable alternative to standard learning management systems.

I am thus very interested in your experiences with the eduComponents. Please fill out the short questionnaire below and send it back to me. Your answers are highly relevant for my research and your feedback will be highly valued. Your information will be used as input for further research and development and will be treated as confidential.

Many thanks in advance

Michael Piotrowski

---

1. Your institution or company: _____

2. Your role with respect to eduComponents (check all that apply):

   - ☐ Instructor
   - ☐ E-learning consultant
   - ☐ System administrator
   - ☐ Software developer

1. Message-ID: `<x6zlpquwrl.fsf@eurus.zrh.dynalabs.de>`
2. `educomponents@uni-magdeburg.de`

3. If you are an instructor, please describe your usage scenario (including course types and subjects, number of students, primary type of instruction, i.e., distance/blended/face-to-face learning):

4. Which of the eduComponents products are you using? Check all that apply:

   - ☐ ECLecture
   - ☐ ECQuiz
   - ☐ ECAssignmentBox
   - ☐ ECAutoAssessmentBox and ECSpooler
   - ☐ ECReviewBox

5. Are there any specific reasons or requirements why you are using eduComponents? _____

6. You are using Plone as a platform for the eduComponents. Are you also using Plone as a content management system for Web pages? _____

7. Do you use other software packages for e-learning (such as a learning management system, a collaboration platform, a teleconferencing software)? If yes, please name them: _____

8. What do you like best about the eduComponents? Do you have any suggestions or comments? _____

9. May I contact you for more specific questions? _____

Bibliography

[1] *SCORM 2004*. Advanced Distributed Learning, 3rd edition, 2006. URL http://adlnet.gov/ (accessed 2008-10-17).

[2] Mario Amelung. *Web-Services für das E-Assessment*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, Magdeburg, in progress.

[3] Lorin W. Anderson and David R. Krathwohl, editors. *A Taxonomy for Learning, Teaching, and Assessing. A Revision of Bloom's Taxonomy of Educational Objectives*. Allyn & Bacon, Boston, MA, USA, 2nd edition, 2000.

[4] Patti Anklam. The Camelot of collaboration: The case of VAX Notes. *Knowledge Management Magazine*, 5(2), 2001. URL http://www.byeday.net/assets/documents/Camelot%20of%20Collaboration%20Patti%20Anklam.pdf (accessed 2008-10-17).

[5] Anonymous. Europe's e-learning industry group revamped, renamed. *Chief Learning Officer magazine*, May 2007. URL http://www.clomedia.com/newsletters/2007/May/1814/ (accessed 2008-10-17).

[6] Wolfgang Appelt. WWW based collaboration with the BSCW system. In *SOFSEM '99: Proceedings of the 26th Conference on Current Trends in Theory and Practice of Informatics*, number 1725 in Lecture Notes in Computer Science, pages 66–78, Berlin/Heidelberg, 1999. Springer.

[7] Alexander W. Astin. *Assessment For Excellence: The Philosophy And Practice Of Assessment And Evaluation In Higher Education*. American Council on Education Oryx Press Series on Higher Education. Oryx Press, January 1991. doi: 10.1336/1573565512.

[8] Paris Avgeriou, Simos Retalis, and Manolis Skordalakis. An architecture for open learning management systems. In *Advances in Informatics*, volume 2563 of *Lecture Notes in Computer*

Science, pages 83–99. Springer, Berlin/Heidelberg, 2003. doi: 10.1007/3-540-38076-0_13.

[9] R. A. Avner and Paul Tenczar. The TUTOR manual. CERL Report X-4, University of Illinois, Urbana, IL, 1969. ERIC ED050583.

[10] Helen C. Barrett. Researching electronic portfolios and learner engagement. White paper, REFLECT Initiative, 2005. URL `http://electronicportfolios.org/reflect/whitepaper.pdf` (accessed 2008-10-17).

[11] Tony Bates. *National strategies for e-learning in post-secondary education and training*. United Nations Educational, Scientific and Cultural Organization (UNESCO), Paris, 2001. URL `http://unesdoc.unesco.org/ulis/cgi-bin/ulis.pl?catno=126230` (accessed 2008-10-17).

[12] Peter Baumgartner. Förderprogramm Neue Medien in der Bildung: Audit-Bericht des Experten/innen-Teams unter Vorsitz von Prof. Dr. Peter Baumgartner. Technical report, DLR Projektträger Neue Medien in der Bildung und Fachinformation, St. Augustin, December 2003. URL `http://www.dlr.de/pt_nmb/Foerderung/Bekanntmachungen/Audit_Bericht_2003.pdf` (accessed 2008-10-17).

[13] Peter Baumgartner, Hartmut Häfele, and Kornelia Maier-Häfele. *E-Learning Praxishandbuch: Auswahl von Lernplattformen. Marktübersicht – Funktionen – Fachbegriffe.* StudienVerlag, Innsbruck, 2002.

[14] Peter Baumgartner, Hartmut Häfele, and Kornelia Meier-Häfele. *Content Management Systeme in e-Education*. StudienVerlag, Innsbruck, 2004.

[15] Randy Elliot Bennett, William C. Ward, Donald A. Rock, and Colleen LaHart. Toward a framework for constructed-response items. Technical report, Educational Testing Service, Princeton, NJ, USA, June 1990. ERIC ED395032.

[16] Stefan Bergstedt, Stefan Wiegreffe, Jochen Wittmann, and Dietmar Möller. Content management systems and e-learning systems – a symbiosis? In *Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT'03)*, pages 155–159. IEEE, 2003. doi: 10.1109/ICALT.2003.1215047.

[17] Tim Berners-Lee and Dan Connolly. Hypertext markup language – 2.0. RFC 1866, Internet Engineering Task Force, 1995. URL `http://ietf.org/rfc/rfc1866.txt` (accessed 2008-10-17).

[18] Donald L. Bitzer, Peter G. Braunfeld, and W. W. Lichtenberger. PLATO: An automatic teaching device. *IRE Transactions on Education*, 4:157–161, 1961. doi: 10.1109/TE.1961.4322215.

[19] Donald L. Bitzer, H. Gene Slottow, and Robert H. Willson. A gaseous discharge display and memory mechanism. United States Patent 3 559 190, January 1971. Filing Date: December 22, 1966.

[20] Zuzana Bizoňová and Daniel Ranc. Model driven LMS platform integration. In *The Third Advanced International Conference on Telecommunications, 2007 (AICT'07)*, pages 25–29. IEEE Computer Society, 2007. doi: 10.1109/AICT.2007.29.

[21] Benjamin S. Bloom, editor. *Taxonomy of Educational Objectives, Handbook 1: Cognitive Domain*. Longman, White Plains, NY, USA, 1956.

[22] Scott Bradner. The Internet standards process – revision 3. RFC 2026 (BCP 9), Internet Engineering Task Force, 1996. URL `http://ietf.org/html/rfc2026` (accessed 2008-10-17).

[23] Frederick P. Brooks, Jr. *The Mythical Man-Month*. Addison-Wesley, Harlow, UK, 2nd edition, 1995.

[24] Gavin Busuttil-Reynaud and John Winkley. e-Assessment Glossary (Extended). Version 1.1, Joint Information Systems Committee (JISC) and Qualifications and Curriculum Authority (QCA), January 2006. URL `http://www.jisc.ac.uk/assessment` (accessed 2008-10-17).

[25] Philippa Butler. A review of the literature on portfolios and electronic portfolios. Project report, eCDF ePortfolio Project, October 2006. URL `https://eduforge.org/docman/view.php/176/1111/` (accessed 2008-10-17).

[26] Viviane Cantaluppi. Analyse des Lernerfolges bei Online-Übungen. Diplomarbeit, Universität Zürich, 2007. URL `http://www.ifi.uzh.ch/teaching/studiengaenge/allg_infos/diplomarbeiten/abstracts_2007/cantaluppi_viviane/` (accessed 2008-10-17).

[27] Susan M. Case and David B. Swanson. *Constructing Written Test Questions For the Basic and Clinical Sciences*. National Board of Medical Examiners, Philadelphia, PA, USA, 3rd edition, 2002. URL `http://www.nbme.org/PDF/ItemWriting_2003/2003IWGwhole.pdf` (accessed 2008-10-17).

[28] Catalyst IT Ltd. Technical evaluation of selected learning management systems. Technical report, The Open Polytechnic of New Zealand, Wellington, New Zealand, May 2004. URL `https://eduforge.org/docman/view.php/7/18/` (accessed 2008-10-17).

[29] Lorinda Cherry. Computer aids for writers. *ACM SIGOA Newsletter*, 2(1–2):61–67, 1981. doi: 10.1145/1159890.806455.

[30] Arthur W. Chickering and Zelda F. Gamson. Seven principles for good practice in undergraduate education. *AAHE Bulletin*, 39(7): 3–7, March 1987.

[31] Donald Chinn. Peer assessment in the algorithms course. In *ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, pages 69–73, New York, NY, USA, 2005. ACM Press. doi: 10.1145/1067445.1067468.

[32] Koen Claessen and John Hughes. QuickCheck: a lightweight tool for random testing of haskell programs. In *ICFP '00: Proceedings of the fifth ACM SIGPLAN international conference on Functional programming*, pages 268–279, New York, NY, USA, 2000. ACM Press. doi: 10.1145/351240.351266.

[33] Michael D. Cohen, James G. March, and Johan P. Olsen. A garbage can model of organizational choice. *Administrative Science Quarterly*, 17(1):1–25, March 1972.

[34] Betty Collis and Jef Moonen. *Flexible Learning in a Digital World: Experiences and Expectations*. Open & Distance Learning Series. Kogan Page, London, UK, April 2001.

[35] Betty Collis and Marijk van der Wende. Models of technology and change in higher education. Report, Center for Higher Education Policy Studies, University of Twente, Enschede, The Netherlands, December 2002. URL `http://purl.org/utwente/44610` (accessed 2008-10-17).

[36] Gráinne Conole, Janice Smith, and Su White. A critique of the impact of policy and funding. In Gráinne Conole and Martin Oliver, editors, *Contemporary Perspectives in E-Learning Research: Themes, methods and impact on practice*, Open and Flexible Learning Series, chapter 3, pages 38–54. Routledge, Abingdon, UK, 2007.

[37] Declan Dagger, Alexander O'Connor, Séamus Lawless, Eddie Walsh, and Vincent P. Wade. Service-oriented e-learning platforms: From monolithic systems to flexible services. *IEEE Internet Computing*, 11(3):28–35, May/June 2007. doi: 10.1109/MIC.2007.70.

[38] Charlie Daly and Jane Horgan. Patterns of plagiarism. In *SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education*, pages 383–387, New York, NY, USA, 2005. ACM. doi: 10.1145/1047344.1047473.

[39] Will M. Davies and Hugh C. Davis. Designing assessment tools in a service oriented architecture. In Giovanna Albano, Pierluigi Ritrovato, and Saverio Salerno, editors, *1st International ELeGI Conference on Advanced Technology for Enhanced Learning*, Vico Equense, Italy, March 2005. URL `http://www.bcs.org/upload/pdf/ewic_el05_slpaper4.pdf` (accessed 2008-10-24).

[40] Brent Davis, Dennis Sumara, and Rebecca Luce-Kapler. *Engaging Minds*. Lawrence Erlbaum Associates, Mahwah, NJ, USA, June 2000.

[41] Willem de Boer. *Flexibility support for a changing university*. PhD thesis, University of Twente, Enschede, The Netherlands, 2004. URL `http://purl.org/utwente/41410` (accessed 2008-10-17).

[42] Christian Dervaric. Erkennung und Behandlung von Plagiaten bei Lösungen zu Übungsaufgaben. Diplomarbeit, Fakultät für Informatik, Otto-von-Guericke-Universität, Magdeburg, 2007.

[43] F. Di Domenico, Emanuele Panizzi, Andrea Sterbini, and Marco Temperini. Analysis of commercial and experimental e-learning sytems. Technical report, TISIP Research Foundation, Trondheim, Norway, 2005. URL http://www2.tisip.no/quis/public_files/wp4-analysis-of-e-systems.pdf (accessed 2008-10-17).

[44] Horst Dichanz and Annette Ernst. E-Learning: Begriffliche, psychologische und didaktische Überlegungen zum "electronic learning". *MedienPädagogik*, 0(2), 2001. URL http://www.medienpaed.com/00-2/dichanz_ernst1.pdf (accessed 2008-10-17).

[45] Walter Dick. The development and current status of computer-based instruction. *American Educational Research Journal*, 2(1): 41–54, 1965. doi: 10.2307/1162068.

[46] Mahesh H. Dodani. The dark side of object learning: Learning objects. *Journal of Object Technology*, 1(5):37–42, November/December 2002. URL http://www.jot.fm/issues/issue_2002_11/column3 (accessed 2008-10-17).

[47] Christopher Douce, David Livingstone, and James Orwell. Automatic test-based assessment of programming: A review. *Journal on Educational Resources in Computing*, 5(3):4, 2005. doi: 10.1145/1163405.1163409.

[48] Stephen Downes. Resource profiles. *Journal of Interactive Media in Education*, 2004(5), May 2004. URL http://www-jime.open.ac.uk/2004/5/ (accessed 2008-10-17).

[49] Erik Duval, Eddy Forte, Kris Cardinaels, Bart Verhoeven, Rafael Van Durm, Koen Hendrikx, Maria W. Forte, Norbert Ebel, Maciej Macowicz, Ken Warkentyne, and Florence Haenni. The ARIADNE knowledge pool system. *Commun. ACM*, 44(5):72–78, May 2001. doi: 10.1145/374308.374346.

[50] Truls Fagerberg and Torstein Rekkedal. Enhancing the flexibility of distance education: Designing and trying out a learning environment for mobile distance learners. In David Murphy, Ronnie Carr, James Taylor, and Wong Tat-meng, editors, *Distance Education and Technology: Issues and Practice*, pages 335–351. Open University of Hong Kong Press, Hong Kong, 2004. URL http://home.nettskolen.com/~torstein/ICDE2004finalpaper.doc (accessed 2008-10-17).

[51] Federal Republic of Germany. Bundesministerium für Bildung und Forschung (Federal Ministry for Education and Research). "eLearning-Dienste für die Wissenschaft". Richtlinien über die Förderung der Entwicklung und Erprobung von Maßnahmen der Strukturentwicklung zur Etablierung von eLearning in der Hochschullehre im Rahmen des Förderschwerpunkts "Neue Medien in der Bildung". Bekanntmachung des Bundesministeriums für Bildung und Forschung, July 2004. URL http://www.bmbf.de/foerderungen/2576.php (accessed 2008-10-17).

[52] Federal Republic of Germany. Bundesministerium für Bildung und Forschung (Federal Ministry for Education and Research). 2005 report on Germany's technological performance: Main statements from the federal government's point of view, 2005. URL http://technologische-leistungsfaehigkeit.de/en/4281.php (accessed 2008-12-09).

[53] Thomas Feustel. Analyse von Texteingaben in einem CAA-Werkzeug zur elektronischen Einreichung und Auswertung von Aufgaben. Master's thesis, Fakultät für Informatik, Otto-von-Guericke-Universität, Magdeburg, 2006.

[54] Charles J. Fillmore. An alternative to checklist theories of meaning. In *Proceedings of the 1st Annual Meeting*, pages 123–131. Berkeley Linguistics Society, 1975.

[55] George E. Forsythe and Niklaus Wirth. Automatic grading programs. *Commun. ACM*, 8(5):275–278, 1965.

[56] Ashley Friedlein. *Web Project Management: Delivering Successful Commercial Web Sites*. Morgan Kaufmann, San Francisco, CA, USA, 2001.

[57] Rocío García-Robles, Josep Blat, Sergio Sayago, Dai Griffiths, Francis Casado, and Juanjo Martinez. Limitations of some eLearning standards for supporting learning. In M. Auer and U. Auer, editors, *Proceedings of the International Conference of Interactive Computer-Aided Learning, ICL 2004, Villach (Austria)*. In cooperation with ACM-Austrian Chapter, and supported by IEEE Education Society, September 2004. URL http://www.iua.upf.es/~dgriffit/papers/garcia-griffiths_villach.pdf (accessed 2008-10-17).

[58] Matthias Gehrke, Matthias Meyer, and Wilhelm Schäfer. Eine Rahmenarchitektur für verteilte Lehr- und Lernsysteme. online, 2003. URL http://campussource.de/projekte/docs/rahmenarchitektur.pdf (accessed 2007-10-04).

[59] Graham Gibbs and Claire Simpson. Conditions under which assessment supports students' learning. *Learning and Teaching in Higher Education*, 5(1):3–31, 2004. URL http://www.glos.ac.uk/shareddata/dms/2B70988BBCD42A03949CB4F3CB78A516.pdf (accessed 2008-10-17).

[60] Sherwin J. Gooch. Cybernetic music system. United States Patent 4 206 675, June 1980. Filing Date: February 28, 1977.

[61] Pierre Gorissen. Quickscan QTI: Usability study of QTI for De Digitale Universiteit, 2003. URL http://www.gorissen.info/Pierre/qti/Quickscan_QTI_UK.pdf (accessed 2008-10-17).

[62] Pierre Gorissen. Quickscan QTI – 2006: Usability study of QTI for De Digitale Universiteit, 2006. URL http://www.gorissen.info/Pierre/qti/Quickscan_QTI_2006.pdf (accessed 2008-10-17).

[63] William D. Graziadei. VICE in REST. In Teresa M. Harrison and Timothy D. Stephen, editors, *Computer Networking and Scholarly Communication in Twenty-First-Century University*, SUNY series in Computer-Mediated Communication, pages 257–276. SUNY Press, Albany, NY, 1996.

[64] Jonathan Grudin. Why CSCW applications fail: problems in the design and evaluation of organization of organizational interfaces. In *CSCW '88: Proceedings of the 1988 ACM conference on Computer-supported cooperative work*, pages 85–93. ACM Press, 1988. doi: 10.1145/62266.62273.

[65] Perry Hanson, Anna Tomecka, and Susan Wawrzaszek. LTS statement on course management systems at Brandeis University. Technical report, Brandeis University, Library & Technology Services, Waltham, MA, USA, April 18 2007. URL `http://latte.brandeis.edu/project/LTS-statement-on-CMS-at-Brandeis.pdf` (accessed 2007-04-19).

[66] Brian Harvey. Learning objects and instructional design. *International Review of Research in Open and Distance Learning*, 6 (2), 2005. URL `http://www.irrodl.org/index.php/irrodl/article/view/227/310` (accessed 2008-10-17).

[67] John Henning. *The Art of Discussion-Based Teaching: Opening Up Conversation in the Classroom*. Routledge, New York, NY, USA and London, UK, 2007.

[68] IEEE. 1484.1-2003 IEEE Standard for Learning Technology—Learning Technology Systems Architecture (LTSA), 2003. URL `http://ieeexplore.ieee.org/servlet/opac?punumber=8897` (accessed 2008-10-23).

[69] IEEE. 1484.12.1-2002 IEEE Standard for Learning Object Metadata, 2002.

[70] *IMS Content Packaging Specification*. IMS Global Learning Consortium, 2004. URL `http://imsglobal.org/content/packaging/` (accessed 2008-10-17). Version 1.1.4.

[71] *IMS Question & Test Interoperability Specification*. IMS Global Learning Consortium, 2005. URL `http://imsglobal.org/question/` (accessed 2008-10-17). Version 2.0 Final Specification.

[72] *IMS Simple Sequencing Specification*. IMS Global Learning Consortium, 2003. URL `http://imsglobal.org/simplesequencing/` (accessed 2008-10-17). Version 1.0.

[73] Initiative VirtusD. Berliner Memorandum "VirtusD Virtuelle Universität Deutschland – E-Learning für eine bessere Bildung an den Hochschulen", January 2007. URL `http://www.cedis.fu-berlin.de/aktuelles/cedis/memorandum_virtusd.html` (accessed 2008-10-17).

[74] Instructional Technology Resource Center. Final evaluation and recommendation report. Technical report, Idaho State University, Pocatello, Idaho, USA, Sprint 2007. URL http://www.isu.edu/itrc/resources/LMS_FINAL_REPORT_MOODLE.pdf (accessed 2008-12-11).

[75] Intrallect Limited. intrallect newsletter. Linlithgow, U.K., Issue 3, November 2005. URL http://intrallect.com/index.php/intrallect/content/download/497/1974/file/intrallectnewslettera3.pdf (accessed 2008-10-17).

[76] ISO (International Organization for Standardization). ISO 15836:2003. Information and documentation — The Dublin Core metadata element set, 2003.

[77] ISO (International Organization for Standardization). ISO/IEC 19757-2:2003. Information technology – Document Schema Definition Language (DSDL) – Part 2: Regular-grammar-based validation – RELAX NG, 2003.

[78] ISO (International Organization for Standardization). ISO 8879:1986(E). Information processing — Text and office systems — Standard Generalized Markup Language (SGML), 1986.

[79] Stefan Jablonski, Ilia Petrov, Christian Meiler, and Udo Mayer. *Guide to Web Application and Platform Architectures*. Springer, Berlin/Heidelberg, November 2004.

[80] Tony Jenkins. The motivation of students of programming. In *ITiCSE '01: Proceedings of the 6th annual conference on Innovation and technology in computer science education*, pages 53–56, New York, NY, USA, 2001. ACM. doi: 10.1145/377435.377472.

[81] Jeong-Hoon Ji, Gyun Woo, and Hwan-Gue Cho. A source code linearization technique for detecting plagiarized programs. In *ITiCSE '07: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, pages 73–77, New York, NY, USA, 2007. ACM. doi: 10.1145/1268784.1268807.

[82] Stephen C. Johnson. Lint, a C program checker. CSTR 65, Bell Laboratories, Murray Hill, NJ, December 1977.

[83] Alex Johnstone. *Effective Practice in Objective Assessment*. LTSN Physical Sciences Centre, 2003. URL http://www.physsci.heacademy.ac.uk/publications/practiceguide/effectivepracticeinobjectiveassessment.pdf (accessed 2008-10-17).

[84] David Jones. The conceptualisation of e-learning: Lessons and implications. *Studies in Learning, Evaluation, Innovation and Development*, 1(1):47–55, October 2004. URL http://sleid.cqu.edu.au/viewarticle.php?id=27 (accessed 2008-10-17).

[85] Mike Joy, Nathan Griffiths, and Russell Boyatt. The BOSS online submission and assessment system. *Journal on Educational Resources in Computing*, 5(3):2, 2005. doi: 10.1145/1163405.1163407.

[86] Eugene Judson and Daiyo Sawada. Learning from past and present: Electronic response systems in college lecture halls. *Journal of Computers in Mathematics and Science Teaching*, 21 (2):167–181, 2002.

[87] Leander Kahney. Hypercard: What could have been, August 2002. URL `http://www.wired.com/gadgets/mac/commentary/cultofmac/2002/08/54370` (accessed 2008-10-17).

[88] Gal A. Kaminka and Milind Tambe. A synergy of agent components: social comparison for failure detection. In *Autonomous Agents '98: Proceedings of the second international conference on Autonomous agents*, pages 459–460, New York, NY, USA, 1998. ACM. doi: 10.1145/280765.280899.

[89] James G. Keramas. The impact of new technologies on our changing global environment. In *OCEANS '95. MTS/IEEE. Challenges of Our Changing Global Environment. Conference Proceedings*, volume 1, pages 242–246, 1995. doi: 10.1109/OCEANS.1995.526779.

[90] Karie L. Kirkpatrick. Opencourseware: An "MIT thing"? *Searcher*, 14(10):53–58, November/December 2006. URL `http://dlist.sir.arizona.edu/1873/01/OpenCourseWare.pdf` (accessed 2008-10-17).

[91] Oleg Kiselyov. SXML specification. *ACM SIGPLAN Notices*, 37 (6):52–58, 2002. doi: 10.1145/571727.571736.

[92] Bernd Kleimann and Klaus Wannemacher. Nachhaltigkeitsstrategien für E-Learning im Hochschulbereich. HIS-Kurzinformation B3/2003, Hochschul-Informations-System, Hannover, 2003.

[93] Bernd Kleimann and Klaus Wannemacher. E-Learning-Strategien deutscher Universitäten: Fallbeispiele aus der Hochschulpraxis. HIS-Kurzinformation B4/2005, Hochschul-Informations-System, Hannover, 2005.

[94] Ann Kovalchick and Kara Dawson, editors. *Education and Technology: An Encyclopedia*. ABC-CLIO, Santa Barbara, CA, USA, 2003.

[95] Herbert Kubicek, Andreas Breiter, Arne Fischer, and Christian Wiedwald. Organisatorische Einbettung von E-Learning an deutschen Hochschulen. Technical report, Institut für Informationsmanagement, Bremen, Germany, 2004. URL `http://www.ifib.de/publikationsdateien/MMKH_Endbericht_2004-05-26.pdf` (accessed 2008-10-17).

[96] V. K. Kumar and James L. Rogers. Instructional uses of the Olin experimental classroom. In *Proceedings of the ACM SIGCSE-SIGCUE technical symposium on Computer science and education*, pages 189–191, New York, NY, USA, 1976. ACM. doi: 10.1145/800107.803472.

[97] Carl Lagoze, Sandy Payette, Edwin Shin, and Chris Wilper. Fedora: an architecture for complex objects and their relationships. *International Journal on Digital Libraries*, 6(2):124–138, April 2006. doi: 10.1007/s00799-005-0130-3.

[98] Benedetto Lepori and Chiara Succi. eLearning in Higher Education: Prospects for Swiss Universities. 2nd Report of the Educational Management in the Swiss Virtual Campus Mandate (EDUM), ICeF – NewMinE, 2003. URL `http://www.newmine.org/NewMinE_ePaper2.pdf` (accessed 2008-10-17).

[99] Bo Leuf and Ward Cunningham. *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley, Harlow, UK, 2001.

[100] Bob Little. Harnessing learning technology to succeed in business. *Industrial and Commercial Training*, 34(2):76–79, 2002. URL `http://www.emeraldinsight.com/Insight/html/Output/Published/EmeraldFullTextArticle/Pdf/0370340206.pdf` (accessed 2008-10-17).

[101] Xiaofei Liu, Abdulmotaleb El Saddik, and Nicolas D. Georganas. An implementable architecture of an e-learning system. In *IEEE Canadian Conference on Electrical and Computer Engineering, 2003 (IEEE CCECE 2003)*, volume 2, pages 717–720, 2003. doi: 10.1109/CCECE.2003.1225995.

[102] LMIS Committee. Learning management system recommendation. Technical report, University of Puget Sound, Tacoma, WA, USA, March 7 2008. URL `http://projects.ups.edu/rwthornton/LMS_Moodle_finaldraft.pdf` (accessed 2008-10-17). Final draft.

[103] Craig Locatis and Hana Al-Nuaim. Interactive technology and authoring tools: A historical review and analysis. *Educational Technology Research and Development*, 47(3):63–75, 1999. doi: 10.1007/BF02299634.

[104] Craig Locatis, Eldon Ullmer, Victor Carr, Richard Banvard, Quang Le, Raulie Lo, and Matthews Williamson. Authoring systems reassessed. *Educational Technology Research and Development*, 40(2):77–82, 1992. doi: 10.1007/BF02297052.

[105] Romans Lukashenko, Vita Graudina, and Janis Grundspenkis. Computer-based plagiarism detection methods and tools: An overview. In Boris Rachev, Angel Smrikarov, and Dimo Dimov, editors, *CompSysTech '07: Proceedings of the 2007 International Conference on Computer Systems and Technologies*, pages 1–6, New York, NY, USA, 2007. ACM. doi: 10.1145/1330598.1330642.

[106] Steven R. Malikowski. Factors relted [*sic*] to breadth of use in course management systems. *The Internet and Higher Education*, 11(2):81–86, 2008. doi: 10.1016/j.iheduc.2008.03.003.

[107] Steven R. Malikowski, Merton E. Thompson, and John G. Theis. A model for research into course management systems: Bridging technology and learning theory. *Journal of*

Educational Computing Research, 36(2):149–173, 2007. doi: 10.2190/1002-1T50-27G2-H3V7.

[108] Robert J. Marzano. A theory-based meta-analysis of research on instruction. Technical report, Mid-continent Regional Educational Laboratory, Aurora, CO, USA, December 1998. ERIC ED427087.

[109] Robert J. Marzano. *Designing a new taxonomy of educational objectives*. Corwin Press, Thousand Oaks, CA, USA, 2000.

[110] Robert J. Marzano and John S. Kendall. *The New Taxonomy of Educational Objectives*. Corwin Press, Thousand Oaks, CA, USA, 2nd edition, 2006.

[111] Robert J. Marzano and John S. Kendall. *Designing and Assessing Educational Objectives: Applying the New Taxonomy*. Corwin Press, Thousand Oaks, CA, USA, 2008.

[112] Robin Mason, Chris Pegler, and Martin Weller. E-portfolios: an assessment tool for online courses. *British Journal of Educational Technology*, 35(6):717–727, 2004. doi: 10.1111/j.1467-8535.2004.00429.x.

[113] John McCarthy. Recursive functions of symbolic expressions and their computation by machine, part I. *Commun. ACM*, 3(4):184–195, April 1960. doi: 10.1145/367177.367199.

[114] Patrick McNeill and Steve Chapman. *Research Methods*. Routledge, London, UK and New York, NY, USA, 3rd edition, 2005.

[115] David M. Merrill, Leston Drake, Mark J. Lacy, and Jean A. Pratt. Reclaiming instructional design. *Educational Technology*, 35(5):5–7, 1996. URL `http://id2.usu.edu/Papers/Reclaiming.PDF` (accessed 2008-10-17).

[116] Jacques Monnard and Rolf Brugger. Web based course platforms: Evaluation report. Technical report, Edutech mandate of the Swiss Virtual Campus program, Freiburg, Switzerland, March 2003. URL `http://edutech.ch/lms/ev2.php` (accessed 2008-10-17).

[117] Glenda Morgan. Faculty use of course management systems. Research study, EDUCAUSE Center for Applied Research, Boulder, CO, USA, 2003. URL `http://educause.edu/ir/library/pdf/ers0302/rs/ers0302w.pdf` (accessed 2008-10-17).

[118] R. L. "Bob" Morgan, Scott Cantor, Steven Carmody, Walter Hoehn, and Ken Klingenstein. Federated security: The Shibboleth approach. *EDUCAUSE Quarterly*, 27(4):12–17, 2004.

[119] Derek Morrison. Clark Kent solutions have super-powers – well sort of! Online article, October 15, 2004. URL `http://www.auricle.org/auriclewp/?p=116` (accessed 2008-10-17).

[120] Richard T. Murphy and Lola Rhea Appeal. Evaluation of the PLATO IV computer-based education system in the community

college. *SIGCUE Outlook*, 12(1):12–28, 1978. doi: 10.1145/963847.963849.

[121] David Nagel. Louisiana State moves to Moodle. *Campus Technology*, November 2 2007. URL http://www.campustechnology.com/article.aspx?aid=52667 (accessed 2008-10-17).

[122] Attila Nagy. e-Learning. E-Content Report 6, ACTeN, 2004. URL http://www.acten.net/uploads/images/423/e-learning.pdf (accessed 2008-10-17).

[123] Filip Neven and Erik Duval. Reusable learning objects: a survey of LOM-based repositories. In *MULTIMEDIA '02: Proceedings of the tenth ACM international conference on Multimedia*, pages 291–294, New York, NY, USA, 2002. ACM Press. doi: 10.1145/641007.641067.

[124] Maish R. Nichani. LCMS = LMS + CMS [RLOs]. Online article, May 2001. URL http://www.elearningpost.com/articles/archives/lcms_lms_cms_rlos/ (accessed 2008-10-17).

[125] Helmut M. Niegemann, Steffi Domagk, Silvia Hessel, Alexandra Hein, Matthias Hupfer, and Annett Zobel. *Kompendium multimediales Lernen*. X.media.press. Springer, Berlin, Heidelberg, 2008.

[126] Sebastian Niezgoda and Thomas P. Way. SNITCH: a software tool for detecting cut and paste plagiarism. In *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pages 51–55, New York, NY, USA, 2006. ACM. doi: 10.1145/1121341.1121359.

[127] Diana G. Oblinger and James L. Oblinger, editors. *Educating the Net Generation*. Educause, 2005. URL http://educause.edu/educatingthenetgen (accessed 2008-10-17).

[128] Commission of the European Communities. The eLearning action plan: Designing tomorrow's education. Communication from the Commission to the Council and the European Parliament COM(2001)172 final, Commission of the European Communities, Brussels, March 2001. URL http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2001:0172:FIN:EN:PDF (accessed 2008-10-17).

[129] Mirko Otto. Ontologien zur semantischen Suche in einem Bestand von Dokumenten. Diplomarbeit, Fakultät für Informatik, Otto-von-Guericke-Universität, Magdeburg, 2006. Advisor: Prof. Dr. Dietmar Rösner.

[130] Roger M. Palay. The structure and use of a test generating system designed to facilitate individually paced instruction. In *Proceedings of the ACM SIGCSE-SIGCUE technical symposium on Computer science and education*, pages 100–103, New York, NY, USA, 1976. ACM. doi: 10.1145/800107.803458.

[131] Mike P. Papazoglou. Service-oriented computing: Concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03)*, pages 3–12. IEEE, 2003. doi: 10.1109/WISE.2003.1254461.

[132] Frederik Paulsson and Mikael Berglund. A service oriented architecture-framework for modularized virtual learning environments. In A. Méndez-Vilas, A. Solano Martín, J. A. Mesa González, and J. Mesa González, editors, *Current Developments in Technology-Assisted Education. Proceedings of the Fourth International Conference on Multimedia and Information and Communication Technologies in Education (M-icte2006)*, volume III, pages 1935–1939, Badajoz, Spain, November 2006. FORMATEX. URL `http://www.formatex.org/micte2006/pdf/1935-1939.pdf` (accessed 2008-10-23).

[133] Fredrik Paulsson. *Modularization of the Learning Architecture: Supporting Learning Theories by Learning Technologies*. PhD thesis, KTH, Stockholm, Sweden, 2008. URL `http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-4712` (accessed 2008-10-23).

[134] Randall G. Pembrook. Some implications of students' attitudes toward a computer-based melodic dictation program. *Journal of Research in Music Education*, 34(2):121–133, 1986. doi: 10.2307/3344740.

[135] Perdisco Pty Limited. E-learning in Australian universities: Opportunities and challenges. Whitepaper, Perdisco Pty Limited, Ultimo NSW, Australia, 2002. URL `http://perdisco.com.au/australianUniWp.asp` (accessed 2008-10-17).

[136] Stephen Petrina. Sidney Pressey and the automation of education, 1924–1934. *Technology and Culture*, 45(2):305–330, April 2004. URL `http://muse.jhu.edu/journals/technology_and_culture/v045/45.2petrina.html` (accessed 2008-10-17).

[137] Michael Piotrowski and Wolfram Fenske. Interoperabilität von elektronischen Tests. In Christian Eibl, Johannes Magenheim, and Martin Wessner, editors, *DeLFI 2007: 5. e-Learning Fachtagung Informatik*, Lecture Notes in Informatics, pages 185–196, Siegen, Germany, September 2007. GI-Verlag.

[138] Pithamber R. Polsani. Use and abuse of reusable learning objects. *Journal of Digital Information*, 3(4), 2003. URL `http://jodi.tamu.edu/Articles/v03/i04/Polsani/` (accessed 2008-10-17).

[139] Philippos Pouyioutas, Maria Poveda, Victoria Kalogerou, and Dmitri Apraxine. The Intertest multiple-choice Web-based software. In *Proceedings of the 4th IASTED International Conference on Web-based Education (WBE 2005), February 21–23, 2005, Grindelwald, Switzerland*, pages 436–441. IASTED, ACTA Press, 2005.

[140] Dave Raggett. A review of the HTML+ document format. Technical report, World Wide Web Consortium, 1994. URL http://www.w3.org/MarkUp/htmlplus_paper/htmlplus.html (accessed 2008-10-17).

[141] Brian K. Reid. Electronic mail of structured documents: representation, transmission, and archiving. In Jacques André, Richard Furuta, and Vincent Quint, editors, *Structured Documents*, Cambridge Series on Electronic Publishing, pages 107–118. Cambridge University Press, Cambridge, 1989.

[142] Gabi Reinmann. Bologna in Zeiten des Web 2.0: Assessment als Gestaltungsfaktor. Arbeitsbericht 16, Universität Augsburg, Augsburg, September 2007. URL http://www.imb-uni-augsburg.de/files/Arbeitsbericht16.pdf (accessed 2008-10-17).

[143] *Respondus User Guide*. Respondus, Inc., Redmond, WA, USA, June 2008. URL http://www.respondus.com/downloads/Respondus35UserGuidePN.doc (accessed 2008-10-17).

[144] Christoph Revermann. eLearning in Forschung, Lehre und Weiterbildung in Deutschland: Sachstandsbericht zum Monitoring eLearning. TAB-Arbeitsbericht 107, Büro für Technikfolgen-Abschätzung beim Deutschen Bundestag (TAB), Berlin, 2006. URL http://www.tab.fzk.de/de/projekt/zusammenfassung/ab107.pdf (accessed 2008-10-17).

[145] Jim Reynolds and Arminder Kaur. Content management. Microsoft enterprise services white paper, Microsoft Corporation, April 2000. URL http://www.microsoft.com/technet/archive/itsolutions/ecommerce/maintain/operate/contmgt.mspx (accessed 2008-10-17).

[146] Randy L. Ribler and Marc Abrams. Using visualization to detect plagiarism in computer science classes. In *INFOVIS '00: Proceedings of the IEEE Symposium on Information Vizualization 2000*, pages 173–177, Washington, DC, USA, 2000. IEEE Computer Society. doi: 10.1109/INFOVIS.2000.10001.

[147] Ronald L. Rivest. S-expressions. Internet draft, Internet Engineering Task Force, 1997. URL http://people.csail.mit.edu/rivest/Sexp.txt (accessed 2008-10-17).

[148] Robby Robson. WWW-based course-support systems: The first generation. *International Journal of Educational Telecommunications*, 5(4):271–282, 1999. URL http://go.editlib.org/p/8835 (accessed 2008-10-17). Journal version of an ED-Media paper: http://www.eduworks.com/Documents/Workshops/Webnet2000/supptsys.html.

[149] Theodor Rütter. *Formen der Testaufgabe: Eine Einführung für didaktische Zwecke*. Beck, München, 1973.

[150] Riku Saikkonen, Lauri Malmi, and Ari Korhonen. Fully automatic assessment of programming exercises. In *ITiCSE '01: Proceedings of the 6th annual conference on Innovation and technol-*

ogy in computer science education, pages 133–136. ACM Press, 2001. doi: 10.1145/377435.377666.

[151] George P. Schell and Max Burns. Merlot: A repository of e-learning objects for higher education. *e-Service Journal*, 1(2):53–64, 2002. URL `http://muse.jhu.edu/journals/eservice_journal/v001/1.2schell.html` (accessed 2008-10-17).

[152] Klaus-Dieter Schewe, Bernhard Thalheim, Aleksander Binemann-Zdanowicz, Roland Kaschek, Thomas Kuss, and Bernd Tschiedel. A conceptual view of Web-based e-learning systems. *Education and Information Technologies*, 10(1–2):83–110, January 2005. doi: 10.1007/s10639-005-6749-2.

[153] Guillaume Schiltz and Andreas Langlotz. eHistling – integrating communication and cooperation in the humanities. In Elspeth McKay, editor, *International Conference on Computers in Education 2004.*, page 149, Altona, Victoria, Australia, 2004. Asia-Pacific Society for Computers in Education (APSCE), Common Ground. Book of Abstracts, full paper on CD-ROM.

[154] Guillaume Schiltz and Andreas Langlotz. The COLAC model: Collaborative paper-writing in the humanities. In Riichiro Mizoguchi, Pierre Dillenbourg, and Zhiting Zhu, editors, *Learning by Effective Utilization of Technologies: Facilitating Intercultural Understanding*, pages 119–122. IOS Press, Amsterdam, The Netherlands, 2006.

[155] Samuel Schluep. *Modularization and structured markup for Web-based learning content in an academic environment*. PhD thesis, ETH Zürich, 2005. URL `http://e-collection.ethbib.ethz.ch/view/eth:28088` (accessed 2008-10-17).

[156] Rolf Schulmeister. *Lernplattformen für das virtuelle Lernen. Evaluation und Didaktik*. Oldenbourg, München, 2002.

[157] Niall Sclater. The demise of eAssessment interoperability? In Hugh Davis, Erik Duval, Brandon Muramatsu, Su White, and Frans Van Assche, editors, *Proceedings of the Workshop on Exchanging Experiences in Technology Enhanced Learning – What Went Wrong? What Went Right? (WWWrong'07)*, volume 317 of *CEUR Workshop Proceedings*, pages 70–74, Sissi, Greece, September 2007. CEUR-WS.org. URL `http://ceur-ws.org/Vol-317/paper07.pdf` (accessed 2008-10-17).

[158] Niall Sclater, Boon Low, and Niall Barr. Interoperability with CAA: does it work in practice? In Myles Danson, editor, *Proceedings of the 6th International Conference on Computer-Assisted Assessment*, pages 317–326. Loughborough University, 2002. doi: 2134/1890.

[159] Peter Scott and Christine Vanoirbeek. Technology-enhanced learning. *ERCIM News*, 2007(71):12–13, October 2007.

[160] Christian Sengstag and Damian Miller. Von der klassischen Vorlesung zur Bologna-kompatiblen Lehrveranstaltung: Redesign einer Lehrveranstaltung. *Zeitschrift für Hochschuldidaktik*, 2005 (4):63–74, June 2005. URL `http://www.zfhd.de/index.php?document_id=1000138&view=set` (accessed 2008-10-17).

[161] Renu Seth. Learning Object Metadata and its application. In *Conference on ICT for Facilitating Digital Learning Environments*, Bangalore, India, January 2006. DRTC. URL `https://drtc.isibang.ac.in/handle/1849/229` (accessed 2008-10-17).

[162] Charles Severance and Joseph Hardin. Strategic directions for sakai and data interoperability. Technical report, Sakai Foundation, June 6 2006. URL `http://www-personal.umich.edu/~csev/papers/2006/2006_07_Roadmap_Interop.pdf` (accessed 2008-10-24).

[163] Bruce Sherwood. Speech synthesis applied to language teaching. *Studies in Language Learning*, 3(1):171–181, 1981. ERIC ED218943.

[164] Miguel-Angel Sicilia, Elena García-Barriocanal, Salvador Sánchez-Alonso, and Jesús Soto. A semantic lifecycle approach to learning object repositories. In Petre Dini, Pascal Lorenz, Mario Freire, Pierre Rolin, Pawel Szulakiewicz, and Tulin Atmaca, editors, *Telecommunications, 2005. Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop. AICT/SAPIR/ELETE 2005. Proceedings*, pages 466–471, Lisbon, July 2005. IEEE Computer Society. doi: 10.1109/AICT.2005.13.

[165] Robert E. Silverman. Auto-instructional devices: Some theoretical and practical considerations. *The Journal of Higher Education*, 31(9):481–486, 1960. doi: 10.2307/1978637.

[166] Jirarat Sitthiworachart and Mike Joy. Effective peer assessment for learning computer programming. In *ITiCSE '04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, pages 122–126, New York, NY, USA, 2004. ACM Press. doi: 10.1145/1007996.1008030.

[167] David Small and Sandy Small. Plato rising. *ANTIC*, 3(3): 36–44, July 1984. URL `http://www.atarimagazines.com/v3n3/platorising.html` (accessed 2008-10-17).

[168] Colin Smythe and P. Roberts. An overview of the IMS Question & Test Interoperability specification. In *Proceedings of the 4th CAA Conference*, Loughborough, England, 2000. Loughborough University. URL `http://hdl.handle.net/2134/1784` (accessed 2008-10-17).

[169] Jack E. Stifle. The PLATO V terminal. CERL Report X-50, University of Illinois, Urbana, IL, April 1978. ERIC ED200244.

[170] Christophe Strobbe. Accessibility issues in two specifications for e-learning tests: IMS QTI 1.2 and IMS QTI 2.0. In *Computers Helping People with Special Needs. 10th International Conference, ICCHP 2006, Linz, Austria, July 11–13, 2006. Proceedings*, volume 4061 of *Lecture Notes in Computer Science*, pages 544–551, Berlin/Heidelberg, 2006. Springer. doi: 10.1007/11788713_81.

[171] Emily Stuart. University to implement Moodle by end of year. *Daily Reveille*, 112(127), April 22 2008. URL `http://www.lsureveille.com/news/1.762154` (accessed 2008-10-17).

[172] Stuart A. Sutton. IEEE 1484 LOM mappings to Dublin Core. Learning Object Metadata: Draft Document v3.6, IEEE Learning Technology Standards Committee (LTSC), 1999. URL `http://www.ischool.washington.edu/sasutton/IEEE1484.html` (accessed 2008-10-17).

[173] Clemens Szyperski, Dominik Gruntz, and Stephan Murer. *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley, Harlow, UK, 2002.

[174] Jo Ann Thrush and Randolph S. Thrush. Microcomputers in foreign language instruction. *The Modern Language Journal*, 68 (1):21–27, 1984. doi: 10.2307/327690.

[175] Andrew Trotter. Blackboard vs. Moodle: Competition in course-management market grows. *Digital Directions*, 2(Spring/Summer):21, June 2008. URL `http://edweek.org/dd/articles/2008/06/09/01moodle.h02.html` (accessed 2008-10-17).

[176] United Kingdom. Department for Education and Skills. *Learning platforms (Secondary)*, 2005. URL `http://publications.teachernet.gov.uk/eOrderingDownload/2102-2005.pdf` (accessed 2008-10-17). Ref. 2102-2005DBW-EN.

[177] Wil van der Aalst and Kees van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, Cambridge, MA, USA, 2004.

[178] Michael Verhaart. Learning object repositories: How useful are they? In Samuel Mann and Tony Clear, editors, *Proceedings of the 17th Annual Conference of the National Advisory Committee on Computer Qualifications (NACCQ)*, pages 458–462, Christchurch, New Zealand, 2004. New Zealand National Advisory Committee on Computing Qualifications. URL `http://www.naccq.ac.nz/conference05/proceedings_04/verhaart-LO.pdf` (accessed 2008-10-17).

[179] Yianna Vovides, Salvador Sanchez-Alonso, Vasiliki Mitropoulou, and Goele Nickmans. The use of e-learning course management systems to support learning strategies and to improve self-regulated learning. *Educational Research Review*, 2(1):64–74, 2007. doi: 10.1016/j.edurev.2007.02.004.

[180] Guijun Wang and Casey K. Fung. Architecture paradigms and their influences and impacts on component-based software systems. In *Proceedings of the 37th Hawaii International Conference on System Sciences*, volume 9, Los Alamitos, CA, USA, 2004. IEEE Computer Society. doi: 10.1109/HICSS.2004.1265643.

[181] Christian Wege. Portal server technology. *IEEE Internet Computing*, 6(3):73–77, 2002. doi: 10.1109/MIC.2002.1003134.

[182] Karl E. Weick. Educational organizations as loosely coupled systems. *Administrative Science Quarterly*, 21(1):1–19, March 1976.

[183] Elizabeth E. Weiner and Lou Ann Emerson. The impact of microcomputers on program excellence. *Journal of Medical Systems*, 10(2):151–162, April 1986. doi: 10.1007/BF00993121.

[184] David A. Wiley. Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In David A. Wiley, editor, *The Instructional Use of Learning Objects: Online Version*, pages 1–35. Agency for Instructional Technology and Association for Educational Communications and Technology, 2000. URL `http://reusability.org/read/chapters/wiley.doc` (accessed 2008-10-17).

[185] David R. Woolley. PLATO: The emergence of online community, 1994. URL `http://thinkofit.com/plato/dwplato.htm` (accessed 2008-10-17).

[186] World Wide Web Consortium. Extensible Markup Language (XML) 1.0, 2006. URL `http://www.w3.org/TR/xml` (accessed 2008-10-17).

[187] World Wide Web Consortium. XML Schema, 2004. URL `http://www.w3.org/XML/Schema` (accessed 2008-10-17).

[188] World Wide Web Consortium. XHTML 1.0 The Extensible HyperText Markup Language, 2002. URL `http://www.w3.org/TR/html` (accessed 2008-10-17).

[189] Kurt D. Zeilenga. Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map. RFC 4510, Internet Engineering Task Force, 2006. URL `http://ietf.org/rfc/rfc4510.txt` (accessed 2008-10-17).

[190] Andreas Zeller. Making students read and review code. In *ITiCSE '00: Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education*, pages 89–92, New York, NY, USA, July 2000. ACM Press.

[191] Heinz Züllighoven and Robert F. Beeger. *Object-Oriented Construction Handbook: Developing Application-oriented Software with the Tools & Materials Approach*. Elsevier, Amsterdam, The Netherlands, 2004.

Author's Publications in the Context of this Dissertation

[P1] Mario Amelung, Michael Piotrowski, and Dietmar Rösner. Webbasierte Dienste für das E-Assessment. In *2. Workshop Pervasive University im Rahmen der GI-Jahrestagung 2007*, Lecture Notes in Informatics, Bremen, Germany, September 2007. GI, GI-Verlag.

[P2] Mario Amelung, Michael Piotrowski, and Dietmar Rösner. Large-scale computer-assisted assessment in computer science education: New possibilities, new questions. In Joseph Fong and Fu L. Wang, editors, *Proceedings of Workshop on Blended Learning 2007*, pages 257–266, Edinburgh, U.K., August 2007. The Hong Kong Web Society. URL `http://www.cs.cityu.edu.hk/~wbl2007/WBL2007_Proceedings_HTML/WBL2007_Proceedings.pdf` (accessed 2008-10-17).

[P3] Mario Amelung, Michael Piotrowski, and Dietmar Rösner. educomponents: A component-based e-learning environment. In *ITiCSE '07: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, pages 352–352, Dundee, U.K., June 2007. ACM SIGCSE, ACM Press. doi: 10.1145/1268784.1268923.

[P4] Michael Piotrowski and Wolfram Fenske. Interoperabilität von elektronischen Tests. In Christian Eibl, Johannes Magenheim, and Martin Wessner, editors, *DeLFI 2007: 5. e-Learning Fachtagung Informatik*, Lecture Notes in Informatics, pages 185–196, Siegen, Germany, September 2007. GI, GI-Verlag.

[P5] Michael Piotrowski, Mario Amelung, and Dietmar Rösner. Tactical, document-oriented e-learning components. In *Proceedings of the IADIS International Conference e-Learning 2007*, Lisbon, Portugal, July 2007. IADIS, IADIS Press.

[P6] Dietmar Rösner, Michael Piotrowski, and Mario Amelung. A sustainable learning environment based on an open source content management system. In Wilhelm Bühler, editor, *Proceedings of the German e-Science Conference (GES 2007)*, Baden-Baden, Germany, May 2007. Helmholtz-Gemeinschaft, Max-Planck-Gesellschaft, and German Rectors' Conference (HRK), Max-Planck-Gesellschaft. URL `http://edoc.mpg.de/315523` (accessed 2008-10-17).

[P7] Michael Piotrowski, Mario Amelung, and Dietmar Rösner. Educomponents: Educational components for Plone. In Miltiadis Lytras and Ambjörn Naeve, editors, *Open Source for Knowledge and Learning Management: Strategies Beyond Tools*. Idea Group, Hershey, PA, USA, 2007.

[P8] Dietmar Rösner, Mario Amelung, and Michael Piotrowski. E-Learning-Komponenten zur Intensivierung der Übungen in der Informatik-Lehre – ein Erfahrungsbericht. In Peter Forbrig, Günter Siegel, and Markus Schneider, editors, *2. GI-Fachtagung Hochschuldidaktik der Informatik*, volume P-100 of *Lecture Notes in Informatics (LNI) – Proceedings*, pages 89–102, Munich, Germany, December 2006. GI, GI-Verlag.

[P9] Mario Amelung, Michael Piotrowski, and Dietmar Rösner. EduComponents: Experiences in e-assessment in computer science education. In *ITiCSE '06: Proceedings of the 11th annual conference on Innovation and technology in computer science education*, pages 88–92, Bologna, Italy, June 2006. ACM SIGCSE, ACM Press. doi: 10.1145/1140124.1140150.

[P10] Michael Piotrowski and Dietmar Rösner. Integration von E-Assessment und Content-Management. In Jörg M. Haake, Ulrike Lucke, and Djamshid Tavangarian, editors, *DeLFI2005: 3. Deutsche e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V.*, volume P-66 of *Lecture Notes in Informatics*, pages 129–140, Rostock, Germany, September 2005. GI, GI-Verlag.

[P11] Dietmar Rösner, Mario Amelung, and Michael Piotrowski. Lls-Checker – ein CAA-System für die Lehre im Bereich Programmiersprachen. In Djamshid Tavangarian Jörg M. Haake, Ulrike Lucke, editor, *DeLFI2005: 3. Deutsche e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V.*, volume P-66 of *Lecture Notes in Informatics*, pages 307–318, Rostock, Germany, September 2005. GI, GI-Verlag.

Theses Supervised by the Author in the Context of this Dissertation

[T1] Christian Dervaric. Erkennung und Behandlung von Plagiaten bei Lösungen zu Übungsaufgaben. Diplomarbeit, Fakultät für Informatik, Otto-von-Guericke-Universität, Magdeburg, 2007. Advisor: Prof. Dr. Dietmar Rösner.

[T2] Thomas Feustel. Analyse von Texteingaben in einem CAA-Werkzeug zur elektronischen Einreichung und Auswertung von Aufgaben. Master's thesis, Fakultät für Informatik, Otto-von-Guericke-Universität, Magdeburg, 2006. Advisor: Prof. Dr. Dietmar Rösner.