EduComponents: Experiences in E-Assessment in Computer Science Education

Mario Amelung
Otto-von-Guericke-Universität
P.O. Box 4120
39016 Magdeburg, Germany
amelung@iws.cs.uni-magdeburg.de

Michael Piotrowski
Otto-von-Guericke-Universität
P.O. Box 4120
39016 Magdeburg, Germany
mxp@iws.cs.uni-magdeburg.de

Dietmar Rösner
Otto-von-Guericke-Universität
P.O. Box 4120
39016 Magdeburg, Germany
roesner@iws.cs.uni-magdeburg.de

ABSTRACT

To reduce the workload of teachers and to improve the effectiveness of face-to-face courses, it is desirable to supplement them with Web-based tools which support the creation, management, submission, and assessment of assignments and tests. This paper presents our approach for supporting computer science education with software components which are integrated into a general-purpose content management system (CMS). We describe the design and implementation of these components, and we report on our practical experience with deploying the software in our courses.

Categories and Subject Descriptors

K.3.1 [**Computer Uses in Education**]: Computer-assisted instruction (CAI); K.3.2 [**Computer and Information Science Education**]: Computer science education

General Terms

Design, Human Factors, Management, Measurement, Performance

Keywords

E-Assessment, Automatic assessment, Student tracking, Content management

1. MOTIVATION

For us, freedom of teaching includes that teachers are liberated from avoidable administrative work, so that they are free to concentrate on teaching and tutoring. For students, learning should not be unnecessarily confined by temporal or local restrictions.

1.1 Problem

Lectures are typically accompanied by exercise courses or tutorials. These courses allow students to review and to apply the knowledge presented in the lecture; they also provide for some monitoring of the students' performance. Exercise courses are therefore an important part of the studies. We felt, however, that many of our exercise courses were inefficient: They offered only relatively little motivation for students and allowed only restricted conclusions about the students' performance during the course, whereas the administration of the courses required a lot of work.

One of the reasons for this situation was that the assignments, which the students handed in on paper, could only be checked selectively. Typically, each assignment was presented on the blackboard by one student in the course, so that teachers only saw a small percentage of the assignments. It was therefore difficult to detect recurring problems and to judge the overall performance and progress of the class.

This situation was also unsatisfactory for students, since their solutions and their problems frequently could not be discussed in detail due to time constraints.

Teachers also had to handle paper forms and had to transfer the data from these forms into the computer to track student performance and to eventually issue course certificates.

The problems were especially grave for programming assignments. Handing in programs on paper and discussing them on the blackboard is only viable for very small programs, and practical problems (e.g., syntax errors) are hard to detect. It is also time-consuming, so only few programming assignments could be handed out. Requiring the submission of programs via e-mail enabled teachers to test the submissions, but it added to their workload, as there was no framework for the management of electronic submissions.

1.2 Goals and approach

We did not want to abolish face-to-face courses, but we wanted to make the courses more efficient and, hopefully, also more effective. To achieve this goal, we envisioned a three-step approach:

1. Complement the traditional, mostly essay-like assignments with regular electronic multiple-choice tests. Multiple-choice tests allow to assess the performance of *all* students of a class without the need for extra grading work for the teacher.

2. Electronic submissions for essay-like assignments and support for the assessment and grading process.

3. Automatic checking and assessment of programming assignments.

On a technical level, the major goal was the central management of tests, assignments, and submissions in a framework covering the whole process: Creation, submission and grading of assignments and tests.

The following sections describe our approach in more detail.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ITiCSE 2006 June 26-28, 2006, Bologna, Italy
Copyright 2006 ACM X-XXXXX-XX-X/XX/XX ...$5.00.

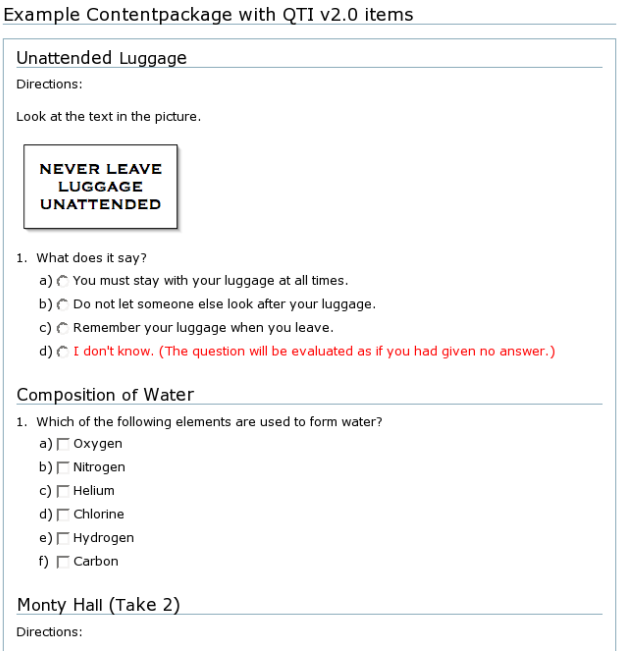

Figure 1: Example multiple choice test in LlsMultipleChoice.

Figure 2: Example assignment box view.

2. EDUCOMPONENTS

We were already using the Plone[1] content management system (CMS) for our Web site, which includes online course material for the courses offered by our research group. It was therefore important to us that all software we would develop to achieve the goals outlined above would fit seamlessly into this framework.

Plone is an open-source Web-based content management system built on top of the Zope[2] Web application framework. Like Plone, Zope is open-source. Zope and Plone are written in Python[3], a widely-used and highly portable open-source programming language. This enables Zope and Plone to run on a 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.

Building on portable open-source software avoids license fees and vendor lock-in, and offers benefits like complete control over the software and the data, the possibility of adapting it to one's specific needs and benefits from a large developer community.

Zope and Plone are extensible through so-called *products*. Products can add new types of content objects to the CMS. They share the same look and feel and they can leverage the facilities provided by the CMS framework, e.g., user and role management, access control and data storage. Custom content types also automatically benefit from Plone features like timed publication, metadata, or indexing and retrieval.

Based on results of our research and on experience gathered with previous partial implementations, we designed, implemented and deployed three Plone products: LlsMultipleChoice, ECAssignmentBox, and ECAutoAssessmentBox. These products provide specialized content types which support the creation and management of assignments and tests, as well as the submission and automated assessment of students' solutions within a general-purpose CMS.

[1] `http://plone.org/`
[2] `http://zope.org/`
[3] `http://python.org/`

2.1 LlsMultipleChoice

LlsMultipleChoice is an extension module for Plone for the creation and delivery of multiple-choice tests (see also [11]). The product supports single-answer as well as multiple-answer 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 possible to present different randomly selected subsets of questions and answers to each student. Figure 1 shows a typical test view.

For self-assessment tests, students can be provided with instant result and/or be allowed to take a test multiple times. Answers can be annotated with explanations which are shown to the candidate in instant-results mode.

Teachers can access detailed reports, providing an overview of the performance of all students. The reports can also be exported for further processing in a spreadsheet or statistics program.

Tests and individual questions and answers can be imported and exported using the IMS QTI v2.0 standard [8].

LlsMultipleChoice is fully internationalized and can easily be adapted to different language environments using Plone's localization facilities. LlsMultipleChoice currently comes with English, German, and French messages.

We're working on support for more test types and more detailed statistical reports.

2.2 ECAssignmentBox

ECAssignmentBox is a Plone product which allows the creation, submission and grading of online assignments (exercises, homework), both for traditional on-site courses and for e-learning.

The basic idea is that teachers write and save assignment texts in a folder-like content object – the *assignment box* – into which students submit their answers or solutions. The submissions are stored as *ECAssignment* objects inside the assignment box. They are then put through a number of *workflow states*, typically 'submitted,' 'accepted,' and 'graded.' Teachers can view the submissions, assign grades and add feedback.

Each assignment box consists of a title and the text of the as-

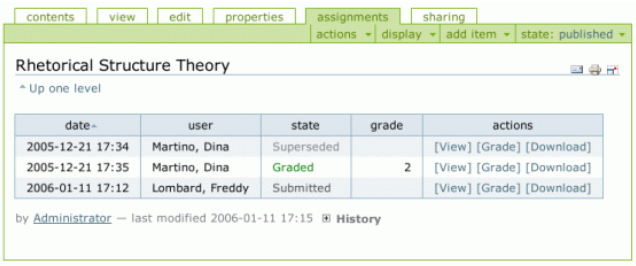

Figure 3: Example assignment box evaluation view.

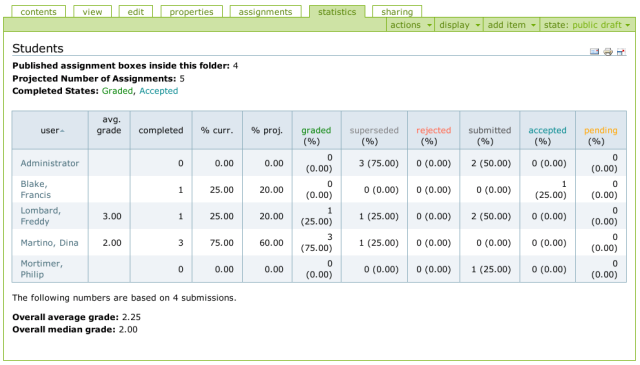

Figure 4: Example ECFolder statistics.

signment. Also, the submission period, i.e., the time frame during which students are allowed to submit their answers, can be specified.

Figure 2 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 until the submission period has ended or until the submission is reviewed.

The assignment box provides an 'assignments' tab, which allows the teacher to see a list of all submissions to this box. Students can use this tab to view their own submissions, including the current workflow state and assigned grades (fig. 3). By selecting a submission, students can view the teacher's feedback.

The assessment of student submissions in ECAssignmentBox is semi-automated, meaning that the teacher does the assessing and is aided by the tool during the entire process of grading students' work and giving feedback. Therefore ECAssignmentBox defines a specialized workflow for student submissions. The workflow is designed to accommodate different ways of handling submissions. The following workflow states are provided [4]:

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.

Furthermore assignment boxes can be grouped together using a special folder (*ECFolder*), which provides a custom view on all included assignment boxes and statistics for all submissions. ECFolders can be nested and can be used to represent, for example, courses and weekly worksheets.

To get an overview of the performance of all students we are using workflow states and grades assigned to student submissions. Teachers can specify the number of planned assignments for a course and the state or states which indicate that a submission is completed. ECAssignmentBox can automatically summarize workflow states and grades for each student's submissions. This way it is possible to show the success of a student based on planned and completed assignments. Also, overall average and median grades are calculated (see fig. 4).

While computer-aided assistance in the areas of instruction, feedback, and student tracking is not new (cf., for example, [2]), ECAssignmentBox is distinguished by its integration into a general-purpose CMS and the application of CMS workflow concepts to assignments.

2.3 ECAutoAssessmentBox

2.3.1 Background

Programming is an essential topic in the introductory courses of computer science curricula. In addition to the more conceptual aspects, programming includes practical aspects as well, e.g., techniques of testing and debugging programs and the use of a programming environment.

It is therefore not surprising that the first systems supporting marking and grading of student solutions for programming exercises were developed and used as soon as 1960 (cf. [5]). Since that time the motivation (among others, large numbers of students) and topics remained relevant: Some of them are security, plagiarism detection, and automatic assessment (cf. [6], [10], and [4]).

CourseMaster (formerly known as Ceilidh) [7] is a widely used system for the automatic assessment of programming exercises. It evaluates student programs on the basis of their output. [5] The tool uses regular expressions to find the substantial values in the output and to compare it with a model solution. Since this can be exhausting and error-prone, Saikkonen et al. developed their own solution for the automatic evaluation of assignments in Scheme, called *Scheme-robo*[13]. Scheme-robo takes advantage of the fact that in functional programming languages programs are functions whose values can be compared directly with the values of the model

[4]The exact meaning of these states is up to the end-users, but we'll describe the typical usage.

[5]It was also tried to evaluate the quality of student solutions using different software metrics and to transfer these metrics to programs in Prolog as well as exercises in object-oriented analysis and object-oriented design. However the latter was not really successful.

solution.

We also utilized this fundamental property of functional languages in our software tool LlsChecker [12]. It was based on a generic architecture, which allowed us to support more than one functional language (Haskell, Scheme, and CommonLisp). However, LlsChecker was not integrated into Plone, which is why we decided to redesign it.

2.3.2 Features and implementation

Starting from our experience with automatic assessment systems for programming exercises, e.g., CourseMaster, TRAKLA2[9], and Kassandra[14], as well as LlsChecker, we designed and implemented ECAutoAssessmentBox.

ECAutoAssessmentBox is a Plone product derived from ECAssignmentBox, i.e., it has the same basic functionality as described above. In addition, ECAutoAssessmentBox provides special support for programming assignments and automatic evaluation of student submissions (currently for Python, Haskell, Scheme, CommonLisp and Prolog). The benefits of automatic assessment of programs are obvious:

- A larger number of programming exercises can be offered so that students get more practical programming experience.
- Feedback is given to the students at any time and any place.
- Teachers can concentrate on tasks which cannot be automated, such as guidance and tutoring.

The automatic assessment of programs is handled by ECSpooler. ECSpooler is a Web service, which, similar to a printer spooler, manages a submission queue and several backends. A backend implements syntax checking and testing for a specific programming language. Backends can use different approaches for testing: For example, we have implemented a backend for Haskell which compares the output of the student solution with the output of a model solution for a set of test data; as an alternative, we have implemented a backend for Haskell which uses QuickCheck [3] for testing based on properties. Backends are derived from general backend classes.

When creating an auto-assignment box, teachers select a backend and specify the data required for automatic testing by the selected backend, e.g., a model solution and test data or QuickCheck properties.

When a student submits a program, it is saved into the auto-assignment box, then passed to ECSpooler, which in turn passes it on to the backend specified by the teacher for this assignment. The results of the tests performed by the backend are immediately returned and displayed, see fig. 5.

Figure 5: Example assignment view.

3. EXPERIENCE

We have been using online multiple-choice tests and automatic testing of programs since winter semester 2003/2004 in various courses offered by our research group. The software used started from initial stand-alone prototypes. It was continuously improved according to the insights gained and has now reached the state described above.

ECAssignmentBox and ECAutoAssessmentBox

Results of examinations in previous years had indicated that a significant number of students tried to avoid solving programming assignments. In the exercise courses they managed to get by with presenting (or rather sketching) solutions for programming tasks on the blackboard that they had copied from other students. The format of these exercise courses obviously did not motivate some students sufficiently to really work on programming tasks.

Students are now required to submit working programs through ECAutoAssessmentBox. This does not only ensure that they work on the programming assignments, but students reported that they are now much more motivated, since they get immediate feedback for their solutions. The motivation is also due to the fact that they know that their submissions are actually reviewed, while previously only a small number of solutions could actually be discussed. Reviewing larger numbers of student submissions is now possible because the submissions are pre-tested and collected at a central location, so that teachers can easily browse and inspect them before the exercise course. Instead of wasting time for copying programs to the blackboard, teachers are now able to discuss specific problems observed in the submitted programs.

Basically the same applies to non-programming assignments and ECAssignmentBox. Students especially value the reporting and statistics features, which help them to track their learning progress, again resulting in better motivation. Furthermore students find it helpful that their assignments are stored centrally, and can quickly accessed for discussion in the course.

Plagiarism in assignments in general and especially in programming assignments is a well-known problem. Copying the solution of a “helpful” peer is easy. So teachers need to think about how they want to deal with this issue. In contrast to the previous system of paper-based exercises, where students could hide behind more commited peers, ECAssignmentBox and ECAutoAssessmentBox offer much better means to cope with the problem. In our experience, the simple knowledge that their teachers can easily review and compare all submissions suffices to discourage many students from plagiarizing.

LlsMultipleChoice

Formative assessment can play a vital role in motivating learners: Learners can see that they are actually acquiring knowledge, it enables them to track their progress and to identify areas where more work is required, and to thereby remain motivated to improve further. It is therefore desirable to expand the use of formative assessment.

Online multiple-choice tests are particularly useful in this context. They can help to make the frequent use of assessments viable through lower costs of deployment and provide greater flexibility with respect to location and timing, and automatic marking allows to give instant feedback.

In contrast to other disciplines (e.g., medicine) where multiple-choice questionnaires are employed regularily, the reputation of multiple-choice tests is poorer in computer science education and the issue often leads to religious debates. Our impression is that this is primarily due to tests that only check for 'knowledge' (and may be simple 'comprehension') in the sense of Bloom's taxonomy [1].

We are employing formative test on a weekly basis in all our lectures and we decisively design these tests to cover higher ranked objectives of Bloom's taxonomy (e.g., 'analysis' or 'evaluation').

To illustrate our approach we take an example from our course in 'document processing'. A chapter in this course deals with XML technology. Analytical thinking can here, for example, be trained and demonstrated when students are exposed to a DTD and have to decide, which of a number of document instances given as answer are valid instances of the DTD. Of course the other way round is equally useful and possible: The question exposes a document instance and the answer set is made up of DTD candidates for this instance. Our experience in general is that once you are as a teacher acquainted with this format of analytical multiple-choice tests the effort to create new tests significantly decreases and is less of a burden.

LlsMultipleChoice has proven to support this use of multiple-choice tests very well. Teachers are enabled to quickly and easily create tests and the reporting features provide a good overview of the students' performance. The ease of creation and deployment allows to use multiple-choice tests more frequently, resulting in a better tracking of the learning progress. However, the acceptance and the effectiveness of multiple-choice tests heavily depends on their design.

4. CONCLUSION

We have presented software components which extend a general-purpose CMS with educational content types for tests, essay-like assignments, and automatically tested programming assignments.

The integration into the CMS has proven to be a good decision, resulting in robust and easy-to-manage products. The CMS provides a uniform look and feel, which makes the products easy-to-use. Students' comments on the products are consistently positive, and teachers report that they have a much better overview of students and assignments, helping them to improve their teaching.

The CMS also serves as *item bank*, a central repository of tests and assignment, enabling the reuse of teaching and learning materials. In the long run, when the repository of learning materials is large enough, it might even be conceivable to create personalized tests and assignments based on the metadata stored with the objects.

5. ACKNOWLEDGMENTS

The work described in this paper was partially supported by the federal state of Saxony-Anhalt, Germany, grant no. 0047M1/0002A.

6. REFERENCES

[1] B. S. Bloom, M. D. Engelhart, E. J. Furst, and W. H. Hill. *Taxonomie von Lernzielen im kognitiven Bereich*. Beltz, Weinheim, 5 edition, 1976.

[2] R. Carver. Computer-Assisted instruction for a first course in computer science. In *Electronic Proceedings for FIE 1996 Conference*. IEEE, 1996.

[3] K. Claessen and J. 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.

[4] C. Daly and J. Waldron. Assessing the assessment of programming ability. In D. Joyce, D. Knox, W. Dann, and T. L. Naps, editors, *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2004, Norfolk, Virginia, USA, March 3-7, 2004*, pages 210–213. ACM, 2004. ISBN 1-58113-798-2.

[5] G. E. Forsythe and N. Wirth. Automatic grading programs. *Commun. ACM*, 8(5):275–278, 1965.

[6] M. Ghosh, B. Verma, and A. Nguyen. An automatic assessment marking and plagiarism detection. In *International Conference on Information Technology and Applications, IT in Engineering: AI, Signal/Image Processing, ICITA02*, pages 274–279, 2002.

[7] C. Higgins, P. Symeonidis, and A. Tsintsifas. The marking system for CourseMaster. In *ITiCSE '02: Proceedings of the 7th annual conference on Innovation and technology in computer science education*, pages 46–50. ACM Press, 2002.

[8] IMS Global Learning Consortium. *IMS Question and Test Interoperability Version 2.0 Final Specification*. 2005.

[9] M. Laakso, T. Salakoski, A. Korhonen, and L. Malmi. Automatic assessment of exercises for algorithms and data structures – a case study with TRAKLA2. In *Proceedings of the 4th Finnish/Baltic Sea Conference on Computer Science Education, October 1-3, 2004, Koli, Finland*, pages 28–36, 2004.

[10] L. Malmi, A. Korhonen, and R. Saikkonen. Experiences in automatic assessment on mass courses and issues for designing virtual courses. In *Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education*, pages 55–59. ACM, 2002.

[11] M. Piotrowski and D. Rösner. Integration von E-Assessment und Content-Management. In D. T. Jörg M. Haake, Ulrike Lucke, editor, *DeLFI2005: 3. Deutsche e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V.*, Lecture Notes in Informatics (LNI) - Proceedings, pages 129–140, Bonn, 2005. GI-Verlag. ISBN 3-88579-395-4; ISSN 1617-5468.

[12] D. Rösner, M. Amelung, and M. Piotrowski. LlsChecker – ein CAA-System für die Lehre im Bereich Programmiersprachen. In D. T. Jörg M. Haake, Ulrike Lucke, editor, *DeLFI2005: 3. Deutsche e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V.*, Lecture Notes in Informatics (LNI) - Proceedings, pages 307–318, Bonn, 2005. GI-Verlag. ISBN 3-88579-395-4; ISSN 1617-5468.

[13] R. Saikkonen, L. Malmi, and A. Korhonen. Fully automatic assessment of programming exercises. In *ITiCSE '01: Proceedings of the 6th annual conference on Innovation and technology in computer science education*, pages 133–136. ACM Press, 2001.

[14] U. von Matt. Kassandra: the automatic grading system. *SIGCUE Outlook*, 22(1):26–40, 1994.