Experiences in Hybrid Learning with eduComponents

Mario Amelung and Dietmar Rösner

Otto-von-Guericke-Universität, FIN/IWS
P.O. Box 4120, 39016 Magdeburg, Germany
{amelung,roesner}@iws.cs.uni-magdeburg.de

Abstract. Since five years we practice hybrid learning in all our courses by combining classroom lectures and group exercises with Web-based e-learning. In this paper we reflect the experiences with our learning environment and discuss the changes in teaching and learning that resulted from the new approach as well as pedagogical concerns and policy issues.

1 Introduction

Before we introduced a hybrid learning environment—made up as a combination from classroom teaching and e-learning—for computer science courses at the University of Magdeburg, we were dissatisfied with some aspects of the traditional way of teaching, practicing and assessing. The traditional way of exercises in a classroom only manner may be sketched as follows:

- design or choose assignments for a weekly exercise sheet according to the state of the course,
- distribute a printed or online PDF version of the exercise sheet,
- students work through the exercise sheet at home,
- in classroom sessions
  - students present their solutions at the blackboard,
  - tutor and peers give (spontaneous) feedback,
  - peers take notes from the presentation and
  - the tutor may take notes about student's performance.

As a variation written submissions may be demanded for that are checked by tutors. But there is always a delay between submission and the reception of the corrected version with comments. For large groups of students manual correction is labor and time intensive.

However, we wished to offer students more detailed discussion on their solutions and problems, more timely feedback, as well as more opportunities to apply their knowledge and to exercise their skills. Especially for programming assignments, the traditional way of handing in programs on paper and discussing them on the blackboard was not very motivating for our students. This approach is only viable for very small programs, and practical problems (e.g., syntax errors) are hard to detect.

In addition, we also wished that teachers were liberated from avoidable (administrative) work and were provided a better overview of the performance and progress of the class.

Now, with the hybrid learning environment based on eduComponents (cf. below) the exercise courses for computer science and programming lectures follow a significantly changed process which may be summarized as follows:

- Design or choose tasks for weekly exercise sheet according to state of the course, additionally take automatic testability into account.
- Make online version of weekly exercise sheet accessible
- Students should work through exercise sheet and
- submit their solutions via an interactive Web interface (using ECAssignmentBox or ECAutoAssessmentBox, cf. below);
- They get immediate feedback and may re-submit improved versions.
- In preparation of the classroom session the tutor gets a complete overview over of the performance of the group as well as of each single student.
- During classroom session students present their solutions using laptop and beamer.
- Peers get online access to all alternative solutions.

In sum: in contrast to a traditional way of teaching which is primarily paper based, our hybrid learning environments makes extensive use of the benefits of electronic documents in combination with the Web.

We have reported about details of eduComponents and the aspects of computer-aided assessment (CAA) elsewhere (e.g., [1], [2], and [3]). In this paper we concentrate on the issues of hybrid learning, i.e., in our case the relation between face-to-face lectures and group exercises and the Web-based e-learning functionalities.

The paper is organized as follows: In section 2 we give an overview of eduComponents, our collection of modules for e-learning and computer-aided assessment. We then describe how eduComponents are employed in our hybrid learning environment. This is followed by a discussion of experiences and lessons learned in section 4. Finally, we summarize our experiences and illustrate some future options for further improving the learning opportunities for students.

2 eduComponents: Design, Implementation and Functionality

Instead of using a separate learning management system (LMS), which would have required additional training and administration, we have chosen a different approach for the hybrid learning environment: A component-based architecture using a general-purpose content management system (CMS) as the basis.

The use of a CMS as foundation for a hybrid learning environment is motivated by the observation that a large percentage of both traditional as well as e-learning is actually document management: Most activities in higher education involve the production, presentation, and review of written material. Thus, instead of re-implementing basic content management functionality, we base our environment on a general-purpose CMS, which provides a reliable implementation of basic document management functionality. In our case this is the open-source content management system *Plone*[1].

[1] `http://plone.org/`

The e-learning-specific functionality is implemented by extension modules for Plone. We have designed, implemented and deployed a number of Plone modules—collectively called eduComponents[2]—that provide specialized content types offering the following main functions (see also [2] and [4]):

- ECLecture: A portal for learning objects, course information and registration.
- ECQuiz: Electronic multiple-choice tests.
- ECAssignmentBox: Electronic submissions for assignments (e.g., essays) and support for the process of assessment and grading.
- ECAutoAssessmentBox: A version of ECAssignmentBox with automatic testing and assessment of assignments with immediate feedback.
- ECReviewBox: An add-on for ECAssignmentBox allowing teachers to create peer review assignments.

These components can be used separately or in combination and, since many basic functions are already provided by the CMS, they already implement much of the standard functionality required in an e-learning environment. If additional features are required, e.g., a discussion forum, bibliographies, a glossary, a Wiki, or domain-specific content types, they can be integrated by adding other Plone modules. The component-based architecture thus makes it easy to create tailor-made learning environments both for pure online learning as well as for hybrid learning scenarios. 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 way, regardless of whether the document is a multiple-choice test or an image. This ensures a consistent and easy-to-learn user interface. The rest of this section describes the individual components in more detail.

2.1 ECLecture

ECLecture is a Plone module for managing lectures, seminars and other courses. ECLecture objects group all course-related information—including course metadata (such as title, instructor, time, location, credits, etc.)—and resources. ECLecture objects can thus serve as a "portal" 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, ECQuiz for tests, or PloneBoard for discussion forums) 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, even on another server. ECLecture also handles the online registration for courses and exercise groups.

2.2 ECQuiz

ECQuiz supports the creation and delivery of multiple-choice tests (see also [5]). Multiple-choice tests are especially useful as formative tests to quickly assess the

[2] All eduComponents modules are freely available as open-source software licensed under the terms of the GNU Public License (cf. `http://wdok.cs.uni-magdeburg.de/software/`).

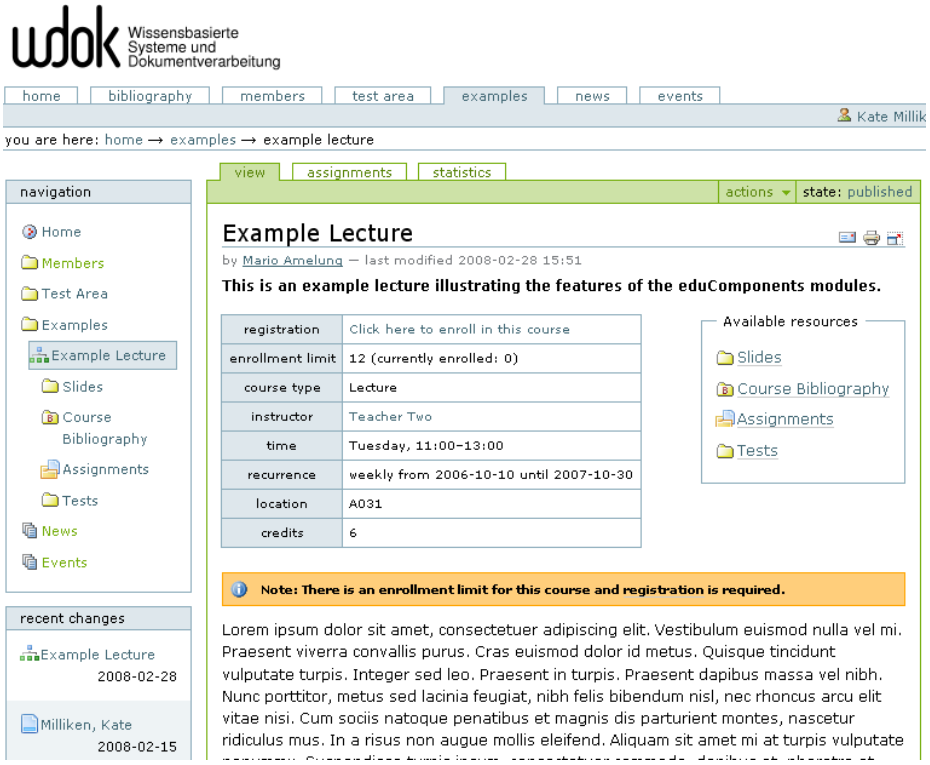

Fig. 1. A typical view of all material from a lecture (realized with ECLecture).

performance of *all* students of a class without the need for extra grading work. ECQuiz also offers tutor-graded extended text questions, so that selected-response and constructed-response items can be mixed in a test to address different skills.

Another possible use of ECQuiz is for self tests with immediate feedback: In this case, students immediately get an overall score, an overview of wrong and right answers and possibly additional explanations, see figure 2. Since both the selection of questions from a pool of questions and the selection of possible answers can be randomized, the instructor may also allow that the test may be taken repeatedly.

2.3 ECAssignmentBox and ECAutoAssessmentBox

ECAssignmentBox supports creation, submission, and grading of essay-like assignments. The assessment of essay-like student submissions offered by ECAssignmentBox is semi-automated, meaning that the teacher does the assessing, but is aided by the tool during the entire process of grading students' work and giving feedback. ECAssignmentBox leverages the workflow capabilities of Plone to define a specialized workflow for student submissions. Modeling the grading process as a workflow structures it and makes it more transparent, but, as in typical content management workflows, it also enables the division

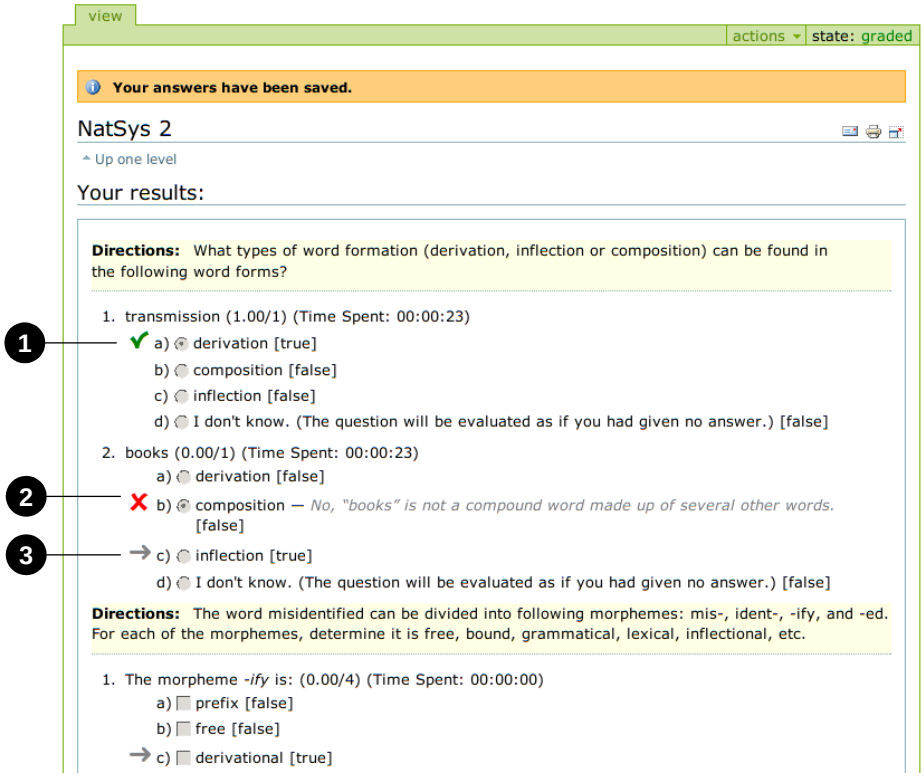

Fig. 2. Example of the ECQuiz instant feedback option for self-assessment tests. ❶ is a correct answer, ❷ is an incorrect answer with additional feedback provided by the test author, and the arrow ❸ indicates the correct answer that the candidate should have selected.

of labor and online collaboration. For example, the detailed reviewing of submissions may be assigned to teaching assistants, while the decision about the eventual grades can be reserved to instructors.

ECAutoAssessmentBox is derived from ECAssignmentBox and was originally developed to allow students to submit their solutions for programming assignments via the Web at any time during the submission period and get immediate feedback (see figure 3). Automatic testing and assessment of assignments is handled by a Web-based service which manages a submission queue and several *backends*. Backends are also Web-based services, which encapsulate the testing functions for a specific type of assignments.

The exact testing strategy implemented by a backend depends on the application: For example, when testing programming assignments, the output of a student solution can be compared to that of a model solution for a set of test data, or the assignment can be tested for properties which must be fulfilled by correct programs. Currently implemented are backends for Haskell, Scheme, Erlang, Prolog, Python, and Java. However, with the appropriate backends, the system can also be used to test submissions in other formal

notations or to analyze natural-language assignments (we have already experimented with style checking and keyword spotting [6]).

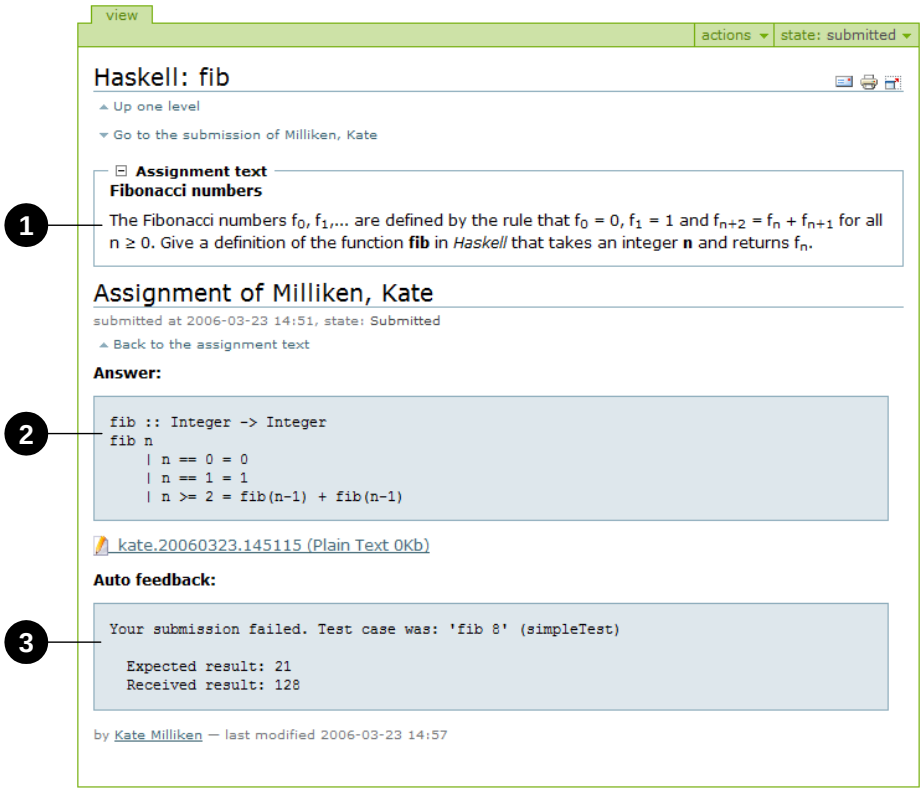

Fig. 3. The view a student gets after the automatic testing of a programming assignment with ECAutoAssessmentBox. ❶ is the assignment; ❷ is the student's submitted program, in this case an incorrect solution; ❸ is the automatic feedback, reporting an error, since the submitted solution does not yield the expected results.

Both ECAssignmentBox and ECAutoAssessmentBox objects represent single assignments. Online exercise sheets are simply created by placing the desired assignments in a special folder, which handles the presentation and provides statistics and analysis features for the student submissions for the contained assignments.

3 Employing eduComponents for Hybrid Learning

In Magdeburg, we are using the eduComponents modules (cf. section 2) as part of a hybrid learning strategy which consists of lectures, electronic exercise work and exercise groups as regular classroom sessions.

We are actively using eduComponents in **all** our lectures since several semesters. This includes advanced lectures in programming like “AI programming and knowledge representation” or “Functional programming: advanced topics” as well as lectures like “Natural language systems”, “Document processing”, or “Information extraction”. In the latter student assignments deal with formal systems and formalisms beyond traditional programming, e.g., XSLT [3] in “Document processing” and regular expressions or UIMA annotators [4] in ’“Information extraction”.

The most recent and broad scale usage in programming was summer semester 2007 in a lecture called “Programming paradigms” with 2 hours lecture and 2 hours exercise per week. This course is obligatory for all CS bachelor students in their second semester. These students have already a background in Java programming from their introductory courses in the first semester and have thus been exposed to the imperative and the object oriented paradigm. Therefore “Programming paradigms” concentrates on complementing the students’ perspective with functional programming (using Haskell, Scheme, and CommonLisp) and logic programming (using Prolog). It is essential that the students deepen their understanding by solving programming tasks in the different representatives of declarative languages. This can only be achieved when exercises and practice are very intensive. We therefore demand that students submit weekly programming assignment several hours prior to the weekly group meeting and get them checked and—hopefully—accepted by ECAutoAssessmentBox. This may involve a number of error corrections and re-submissions.

A group comprises approx. 15 to 20 students and is headed by an assistant as tutor. The tutor of the exercises—this is either an assistant or an advanced student—has prior access to all submissions. He can thus preview all submissions of his group members and look for recurring problems or alternative and outstanding solutions. This allows to better prepare the face-to-face group meetings. The tutor can now decide much better in advance how much time needs to be allocated for what tasks because he can judge the students’ performance and their potential problems from the inspection of submitted solutions and solution attempts. During and after the group session all these documents are available online.

Students have to present their solutions as before, but during the classroom session, the assignments and the presented solutions are projected for all to see. This removes the need to copy solutions to and from the blackboard. The selected students have to comment on their solutions. If the solution and the presentation have been satisfactory, the submission gets moved to the corresponding workflow state (e.g., *accepted* or *graded*).

4 Experiences and Lessons Learned

We started to develop and exploit eduComponents as basis for our hybrid learning environment in winter semester 2003/2004. Since then we have been gathering experiences with this approach in all our lectures. During the last two semesters, automatic testing of programming assignments (using ECAutoAssessmentBox and backends for Haskell,

[3] http://www.w3.org/Style/XSL/
[4] http://incubator.apache.org/uima/

Scheme, and Prolog) was actively used by a total of over 140 students. This resulted in almost 12.000 automatically tested submissions for about 200 assignments.

In the following we will discuss experiences with the hybrid learning arrangement:

- effects on students,
- effects on teachers,
- feedback from students,
- analysis of learning outcomes,
- stipulation of pedagogical experimentation.

4.1 Effects on Students

For programming assignments with automatic testing the demands for students' solutions are much more explicit and rigid with respect to correctness and quality. Students thus also have to ensure that their solution is working correctly. Consequently the intensity of work needed for the exercises has effectively increased.

On the other hand, students can gain access to a larger number of alternative solutions and to typical error cases. Students also reported that they feel much more motivated, since they get immediate feedback for their solutions. The motivation is also due to the fact that students know that their submissions are actually reviewed, while previously only a small number of solutions could be discussed. Maybe these advantages have compensated for the higher requirements.

Student behavior during classroom sessions has also changed: Many students no longer carry written notes to the classroom session, since they know that their submissions are available online. Some students were even tempted not to come at all to the group sessions if their submissions had passed the automatic tests. Our policy in this respect is that personal attendance and active participation in the discussion among peers is obligatory.

A very positive development is that many more students than before speak up in the groups and want to show and discuss their solution if it is different from other presented solutions.

4.2 Effects on Teachers

For teachers using automatic testing of programs, the most significant effect is that the effort for initially designing assignments has increased. This is an insight that other users of automated program testing systems have also reported (e.g., [7]). Automatic testing requires problems and tasks to be formulated much more formally and precisely. This is necessary to enable automatic testing and in order to avoid misunderstandings which could result in students trying to solve a different problem than the one the teacher had in mind and then getting puzzled about the reactions of the automatic testing system.

When they employ eduComponents, teachers are sometimes surprised by unexpected or unintended usage of the system by the students. The latter may demand for policy decisions.

Unexpected usage: ECAssignmentBox has been designed and implemented as lightweight solution. It was intended to support either direct typing of (short) answers or uploading of assignments (programs, texts) from a file; but it intentionally do not offer any sophisticated editor functionality. Nevertheless there were unanticipated usages of the system. Some students used it not only for the submission of their final solution, but also as a kind of "ubiquitous work place" to work on essay-like assignments: They started to work on an assignment from one computer, used the submission feature to store an intermediate version, and later continued to work on the same assignment from a different computer. This resulted in a large number of spurious superseded submissions.

Unintended usage: Other students abused ECAutoAssessmentBox as a Web-based interpreter to solve programming assignments. This was clearly unintended in our design. We therefore introduced a parameter for teachers to restrict the number of possible resubmissions for automatically tested programming assignments. We currently use a limit of three trials. Limiting the misuse of ECAutoAssessmentBox as a trial-and-error device by setting a limit on repeated submissions also enforces a secondary learning objective: We expect that our students are able to use the native programming environments and interpreters for the various programming languages and to leverage them instead of submitting untested sketches of a solution.

4.3 Feedback from Students

At the end of each semester we ask our students to complete a questionnaire on their experience with the learning environment. The questions cover three areas: The use of electronic submissions in general, their effect on the students' working habits, and the usability of eduComponents. The results—based on feedback from up to now more than 200 students—in all three areas are consistently very positive. 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 be accessed for discussion in the course. Students also report that they work more diligently on their assignments because the teachers can now access and review all assignments.

A seemingly minor change in the organization and technical basis of exercises—i.e., introducing that all assignments and all solutions of students are electronic documents in a content management system—resulted in significant changes in the learning environment and changed learning processes much more fundamentally than expected in the beginning of the transition to the new system. When we started using CAA and other e-learning components we had the primary motivation to relief teachers and students from administrative burdens by automating certain processes and supporting others. Our experience is, however, that the change in the way how assignments are submitted has lead to many other changes in our courses because of the new possibilities offered by the system. But the new opportunities also pose new demands for both teachers and students.

Although the workload for students has increased there is a broad acceptance of the new system and students would welcome its use in other lectures as well. We interpret this as a positive reaction on the new opportunities and as an indication that

students accept the higher intensity of their own engagement because they experience and appreciate an improved return on investment for their learning outcomes.

4.4 Analysis of Learning Outcomes

On the other hand an in depth analysis of recent examination results (lecture on "Programming paradigms" from Summer 2007 with first exam in September 2007 and repetition exam in February 2008; N = 55 students) and the recorded data from the respective exercise groups (e.g., final number of accepted solutions, number of presentations of solutions in the group) reveals that not all students make already the intended use of the offered opportunities for self study. In this analysis we distinguished "minimalist" students from "overperformers". We counted those students as minimalists that achieved just those numbers that were obligatory as prerequisite for access to the final exam (a two hour written exam with tasks similar to those from the exercises). Overperformers in contrast have submitted solutions to all or almost all assignments and additionally were much more active in the exercise groups. Whereas all overperformers from the exercise groups—except one, but he got a "very good" as score in the repetition of the exam—excelled their peers with very good or at least good results in the exam, many of the minimalists showed only poor performance or failed, some even twice (i.e., as well in the repetition exam), and no minimalist excelled positively.

4.5 Pedagogical Innovation

There is another effect of our hybrid learning environment: The availability of eduComponents and the relative ease of extending them with new functionality stipulate pedagogical experimentation. The recent development and first employment of the so called ECReviewBox is a point in case.

Computer science students need not only be able to write programs themselves, they must as well be able to understand and evaluate programs written by others. This is, by the way, an excellent example from CS education for the cognitive learning objective of "evaluation" in Bloom's taxonomy [8]. ECReviewBox is a new module of eduComponents that to a large extent automates a peer review process of student solutions to assignments (cf. fig. 4). We have employed this new functionality for several more complex programming assignments for the first time in summer 2007. First, students have been asked to supply their solutions to the programming task. In the subsequent week, all submitters were eligible for acting as anonymous reviewer and commenter on the anonymized submission of some randomly selected other student. Finally the reviews were made available as feedback to the original authors. Please note that such a type of assignment would hardly be possible in a traditional paper-based environment.

5 Summary and Prospect

Since we started with the hybrid learning environment based on combining classroom lectures and exercises with e-learning and especially computer-aided assessment our teaching has changed significantly.

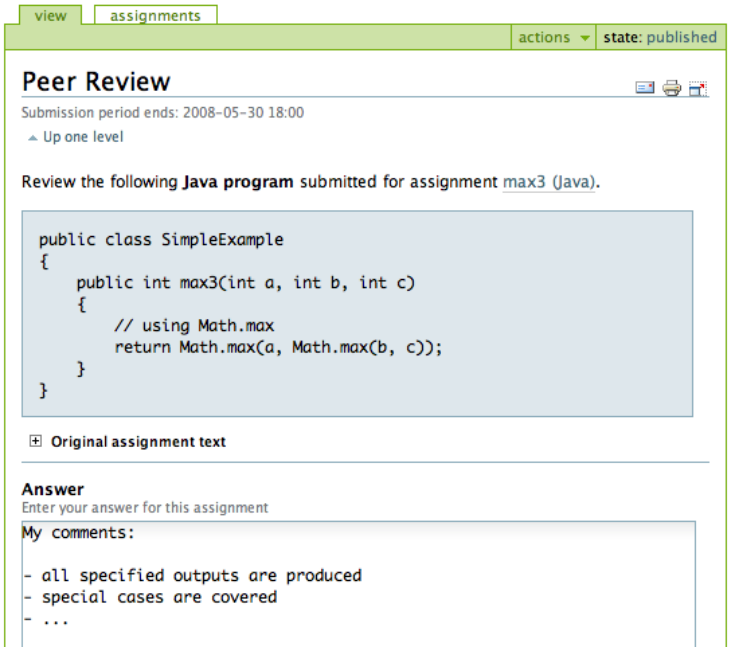

Fig. 4. Review of an assignment using ECReviewBox (student view; simple Java program for illustration purpose).

This starts already when preparing a lecture. When the content is selected and the slides and/or notes are prepared we habitually devise both an accompanying multiple choice questionnaire as well as the assignments for the weekly exercise group. Of course we now do no longer have to create all of this from scratch because we can choose from the repository of former tests and assignments.

The following point illustrates, in our opinion, in a nearly prototypical way the interrelation of pedagogical concerns and policy issues. In summer 2007 we had—as we usually do up to now—demanded that at least 66 percent of the assignments were successfully solved (i.e., accepted by the automatic checking module prior to the exercise groups). We have at several occasions tried to convince our students with arguments that plagiarism is counterproductive and in the long run destructive for themselves. On the other hand we decisively did not regularly check students' submissions for suspected plagiarism (also technically possible within limits), because we did not want to start an "arms race" between plagiarism hiding and plagiarism detection. We now think about a change in policy for future semesters. We will experiment with a combination of relaxing the demands on the one hand (i.e., we will only demand that solutions to a minimum number of assignments have been seriously *attempted*, but they need not necessarily be fully correct and automatically accepted) and on the other hand making use of regular plagiarism detection and rejection of submissions classified as plagiarized. We hope that this change of policy supports the motivating pedagogical concern: Students shall learn

through problem solving on their own not through copying and—hopefully at best—only understanding the solutions of others.

Another policy change that we are planning is to introduce "spontaneous" tasks into the group exercises, i.e., within each group session of 90 minutes a slot of approx. 15 minutes will in the future be allocated for working on problems that are handed to the students within the session. This seems to be both necessary and feasible in the light of our experiences with the hybrid approach of combining e-learning and CAA with a traditional classroom based learning environment.

Acknowledgement

A number of colleagues and students have contributed to the design, implementation and deployment for successful every day usage of the eduComponents. We have to thank our former colleague M. Piotrowski for inspiring discussions and his contributions to the system, our students W. Fenske and S. Peilicke for their substantial implementations and our colleagues I. Blümel, M. Gnjatovic and M. Kunze for their valuable feedback from their experience in teaching with eduComponents.

References

1. Amelung, M., Piotrowski, M., Rösner, D.: 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, New York, ACM Press (2006) 88–92
2. Rösner, D., Piotrowski, M., Amelung, M.: A Sustainable Learning Environment based on an Open Source Content Management System. In Bühler, W., ed.: Proceedings of the German e-Science Conference (GES 2007), Max-Planck-Gesellschaft (2007)
3. Amelung, M., Piotrowski, M., Rösner, D.: 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, New York, NY, USA, ACM Press (2007) 352–352 ISBN:978-1-59593-610-3.
4. Amelung, M., Piotrowski, M., Rösner, D.: Webbasierte Dienste für das E-Assessment. In Koschke, R., Herzog, O., Rödiger, K.H., Ronthaler, M., eds.: Informatik 2007 - Informatik trifft Logistik. Beträge der 37. Jahrestagung der Gesellschaft für Informatik e.V. (GI). Lecture Notes in Informatics, Bonn, GI-Verlag (2007) 518–522 ISBN:978-3-88579-203-1.
5. Piotrowski, M., Rösner, D.: Integration von E-Assessment und Content-Management. In Haake, J.M., Lucke, U., Tavangarian, D., eds.: DeLFI2005: 3. Deutsche e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. Volume P-66 of Lecture Notes in Informatics (LNI)., Bonn, GI-Verlag (2005) 129–140
6. Feustel, T.: 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)
7. Zeller, A.: 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, New York, NY, USA, ACM Press (July 2000) 89–92
8. Bloom, B.S., Engelhart, M.D., Furst, E.J., Hill, W.H., Krathwohl: Taxonomy Of Educational Objectives: Handbook 1, The Cognitive Domain. Allyn & Bacon, Boston (1956)