A Sustainable Learning Environment based on an Open-Source Content Management System

Dietmar Rösner, Michael Piotrowski, Mario Amelung

Otto-von-Guericke-Universität, Institut für Wissens- und Sprachverarbeitung,
Universitätsplatz 2, 39106 Magdeburg, Germany
email: {roesner, mxp, amelung}@iws.cs.uni-magdeburg.de
phone: +49 (391) 67-1 83 43, *fax:* +49 (391) 67-1 20 18

Abstract

This paper presents our approach for supporting face-to-face courses with software components for e-learning based on a general-purpose content management system (CMS). These components—collectively named eduComponents—can be combined with other modules to create tailor-made, sustainable learning environments, which help to make teaching and learning more efficient and effective. We give a short overview of these components, and we report on our practical experiences with the software in our courses.

1 Motivation

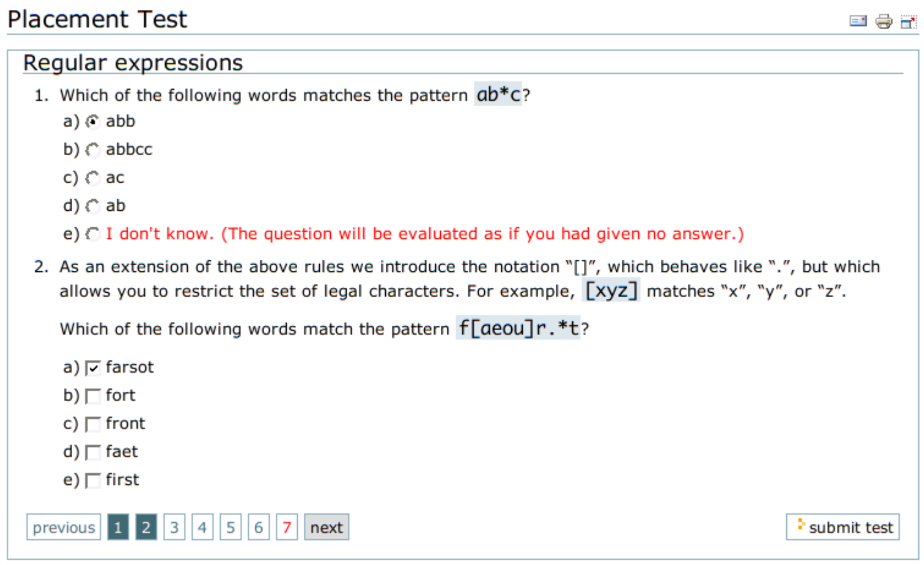
Lectures are typically accompanied by exercise courses or tutorials. These courses are essential for the learning effect: They provide opportunities for students to solidify the knowledge presented in the lecture and to apply it to practical problems. Exercise courses are therefore an important part of university teaching; they also provide teachers with a possibility to monitor the students' performance and learning progress.

We were, however, dissatisfied with a number of aspects of the traditional way of teaching, practicing and assessing in undergraduate computer science courses at our university. For example, we wished to offer students more detailed discussion on their solutions and problems, more timely feedback, as well as more opportunities to apply their new knowledge and to exercise their new skills. Especially for programming assignments, the traditional way of 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.

The desire to make face-to-face courses more efficient and effective was the primary motivation for the development of the e-learning components described in this paper. The new learning environment is designed to make better use of the possibilities of electronic support, computer-aided assessment (CAA) and Web technologies, both for students and for educators.

2 The eduComponents learning environment

Typical learning management systems (LMS), such as Blackboard, ILIAS, Moodle, or OLAT, are specifically designed for e-learning, but are unsuited for general Web content. Thus, at a typical institution, at least two separate systems—an LMS for learning content and a content management system (CMS) for other Web content—have to be installed and administered, and users have to learn the usage of both systems. Typical LMS's also incorporate most common e-learning functions in a single application. However, the conglomeration of various functions results in large and complex systems with high total costs of ownership. It also makes it difficult to adapt these systems to specific needs. All of these aspects negatively affect the acceptance and the sustained use of e-learning.


Placement Test

Regular expressions

1. Which of the following words matches the pattern `ab*c`?
   a) [✓ abb]
   b) abbcc
   c) ac
   d) ab
   e) I don't know. (The question will be evaluated as if you had given no answer.)
2. As an extension of the above rules we introduce the notation "[]", which behaves like ".", but which allows you to restrict the set of legal characters. For example, `[xyz]` matches "x", "y", or "z".

   Which of the following words match the pattern `f[aeou]r.*t`?

   a) [✓ farsot]
   b) fort
   c) front
   d) faet
   e) first

Figure 1: An example multiple-choice test created with ECQuiz. The screenshot shows a single-answer and a multiple-answer question, and the possibility to create multi-page tests (cf. the navigation bar in the lower left). ECQuiz also supports extended text and rating questions, not shown in this screenshot.

Instead of using a separate learning management system (LMS), which would have required additional training and administration, we have chosen a different approach: A component-based architecture using a general-purpose CMS as the basis. The use of a CMS as foundation for an e-learning environment is motivated by the observation that a large percentage of 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 system *Plone*[1].

The e-learning-specific functionality is implemented by extension modules for the CMS. We have designed, implemented and deployed a number of Plone modules—collectively called eduComponents—that provide specialized content types offering the following main functions (see also [1, 7]):

- ECLecture: Course information and registration.
- ECQuiz: Electronic multiple-choice tests.
- ECAssignmentBox: Electronic submissions for essay-like assignments and support for the assessment and grading process.
- ECAutoAssessmentBox: A version of ECAssignmentBox with automatic testing and assessment of assignments with immediate feedback.

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. 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 registration to courses.

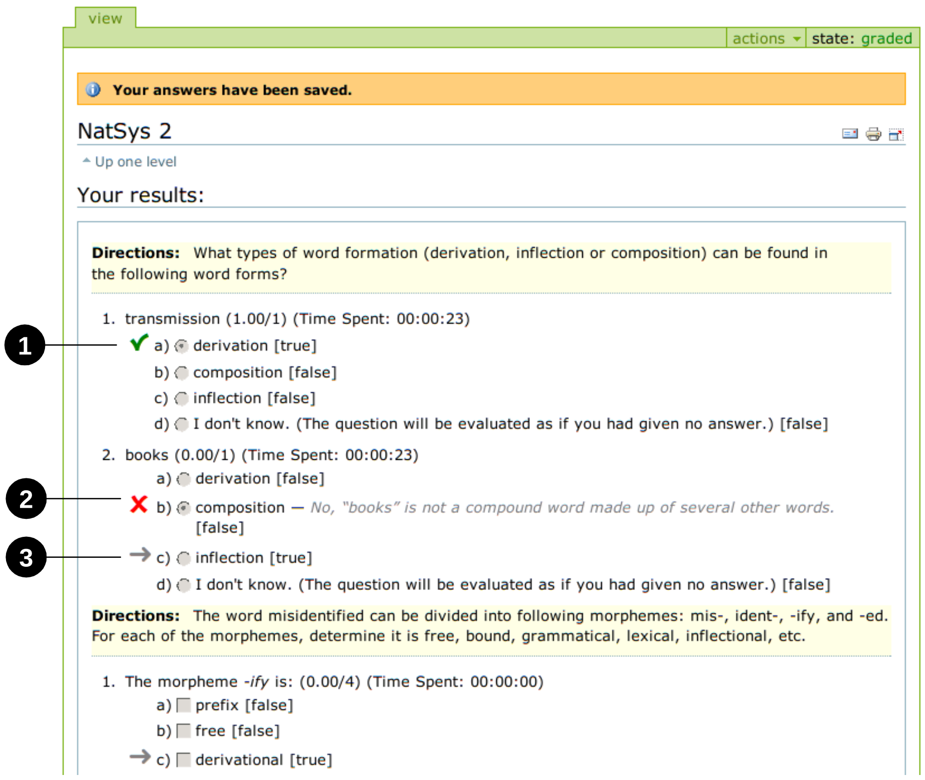
2.2 ECQuiz

ECQuiz (cf. figure 1) supports the creation and delivery of multiple-choice tests (see also [6]). Multiple-choice tests are especially useful as formative tests to quickly assess the performance of *all* students of a class without the need for

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

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.



Your answers have been saved.

NatSys 2

Up one level

Your results:

Directions: What types of word formation (derivation, inflection or composition) can be found in the following word forms?

1. transmission (1.00/1) (Time Spent: 00:00:23)
   ✓ a) ● derivation [true]
   b) ○ composition [false]
   c) ○ inflection [false]
   d) ○ I don't know. (The question will be evaluated as if you had given no answer.) [false]
2. books (0.00/1) (Time Spent: 00:00:23)
   a) ○ derivation [false]
   ✗ b) ● composition — *No, "books" is not a compound word made up of several other words.* [false]
   → c) ○ inflection [true]
   d) ○ I don't know. (The question will be evaluated as if you had given no answer.) [false]

Directions: The word misidentified can be divided into following morphemes: mis-, ident-, -ify, and -ed. For each of the morphemes, determine it is free, bound, grammatical, lexical, inflectional, etc.

1. The morpheme *-ify* is: (0.00/4) (Time Spent: 00:00:00)
   a) ☐ prefix [false]
   b) ☐ free [false]
   → c) ☐ derivational [true]

Figure 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.

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 and 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 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 is 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.

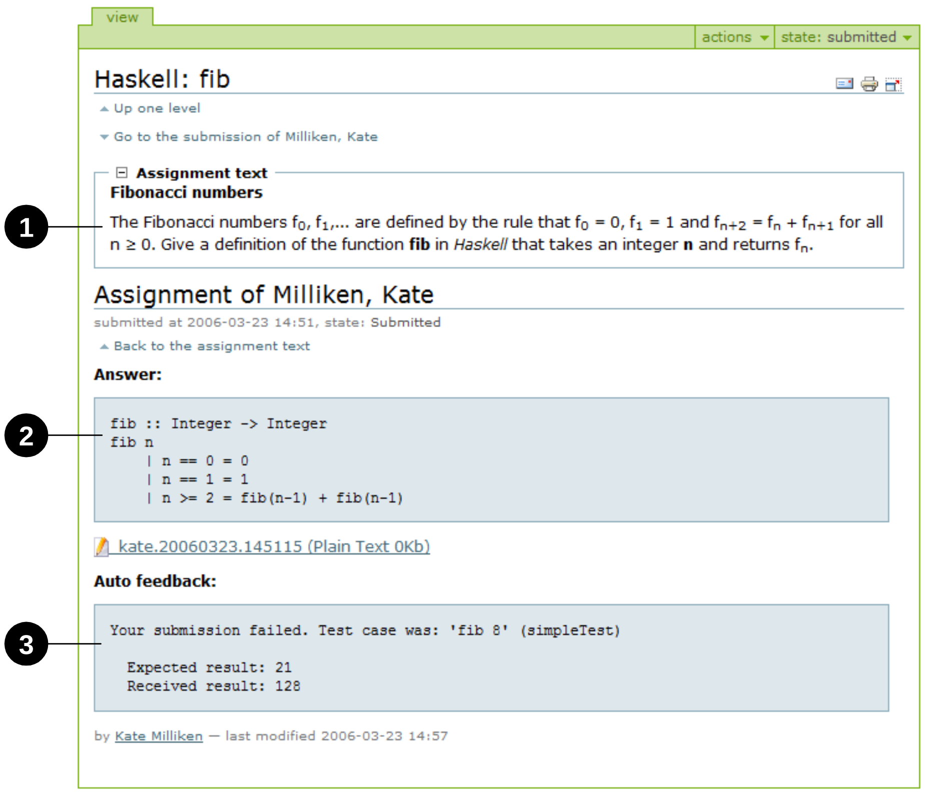
The exact testing strategy implemented by a backend depends on the application: For example, when testing programming assignments, the output of a student solution can be compared to that of a model solution for a set of test data, or the assignment can be tested for properties which must be fulfilled 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 [4]).

Both ECAssignmentBox and ECAutoAssessmentBox objects represent single assignments. Online exercise sheets, as shown in figure 4, 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 Experiences

Since winter semester 2003/2004 we have been gathering experiences with eduComponents for online multiple-choice tests, electronic submission of assignments and automatic testing of programs in our exercise courses. During winter semester 2006/2007 our learning environment was used by over 200 students at our institution.

At the end of each semester we ask our students to complete a questionnaire on their experience with the new e-learning environment. The questions cover three areas: The use of electronic submissions in general, their effect on the



Haskell: fib

Up one level

Go to the submission of Milliken, Kate

Assignment text

Fibonacci numbers

The Fibonacci numbers f_0, f_1,... are defined by the rule that $f_0 = 0$, $f_1 = 1$ and $f_{n+2} = f_n + f_{n+1}$ for all $n \geq 0$. Give a definition of the function **fib** in *Haskell* that takes an integer **n** and returns f_n.

Assignment of Milliken, Kate

submitted at 2006-03-23 14:51, state: Submitted

Back to the assignment text

Answer:

```
fib :: Integer -> Integer
fib n
    | n == 0 = 0
    | n == 1 = 1
    | n >= 2 = fib(n-1) + fib(n-1)
```

kate.20060323.145115 (Plain Text 0Kb)

Auto feedback:

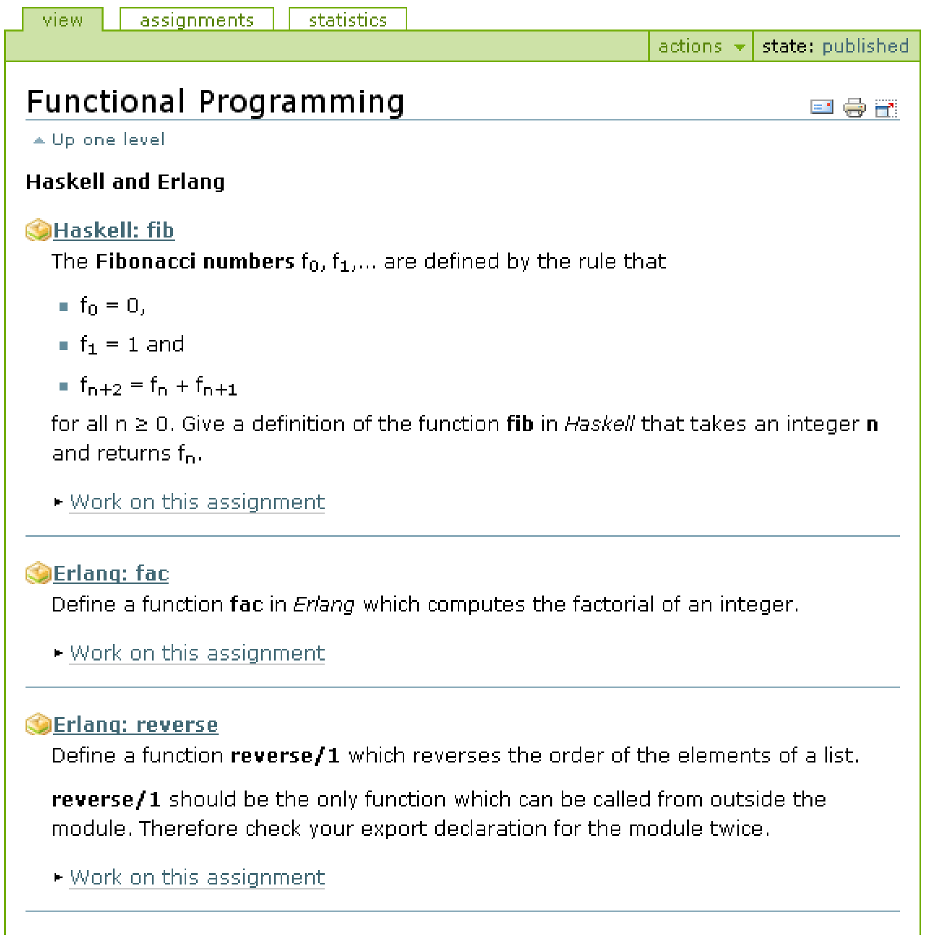
```
Your submission failed. Test case was: 'fib 8' (simpleTest)

  Expected result: 21
  Received result: 128
```

by Kate Milliken — last modified 2006-03-23 14:57

Figure 3: 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.

students' working habits, and the usability of eduComponents. The results in all three areas are consistently very positive.

On the technical side, the integration into the CMS has proven to be a good decision, resulting in a robust system. The CMS provides a uniform look and feel, which makes the learning environment easy-to-use. Students' comments on the usability confirm this, and instructors report that they have a much better overview of students and assignments, helping them to improve their teaching. Students especially value the reporting and statistics features (in fact, a subset of the tools available to instructors), which help them to track their learning progress. Furthermore students find it helpful that their assignments are stored centrally, and can quickly be accessed for discussion in the course. Students



Functional Programming

Up one level

Haskell and Erlang

Haskell: fib

The **Fibonacci numbers** f_0, f_1,... are defined by the rule that

- $f_0 = 0$,
- $f_1 = 1$ and
- $f_{n+2} = f_n + f_{n+1}$

for all $n \geq 0$. Give a definition of the function **fib** in *Haskell* that takes an integer **n** and returns f_n.

Work on this assignment

Erlang: fac

Define a function **fac** in *Erlang* which computes the factorial of an integer.

Work on this assignment

Erlang: reverse

Define a function **reverse/1** which reverses the order of the elements of a list.

reverse/1 should be the only function which can be called from outside the module. Therefore check your export declaration for the module twice.

Work on this assignment

Figure 4: View of a typical online exercise sheet, consisting of several assignment boxes.

also report that they now work more diligently on their assignments because instructors can now easily access and review all assignments.

On the pedagogical side, the processes within the exercise courses have changed much more radically than initially envisaged, especially by the use of ECAssignmentBox and ECAutoAssessmentBox. The traditional learning environment in computer science (and in related subjects) is based on paper and blackboard. Students brought their hand-written notes to the exercise courses and presented their solutions at the board, with their classmates taking notes. This mode of writing and copying was time-consuming and error-prone. Given the time constraints, only a limited number of solutions could be presented and discussed.

In contrast, learning environments realized with eduComponents are based on electronic documents stored and managed in a CMS. Assignments and student submissions can quickly be retrieved, analyzed, annotated, and presented. Previously, student submissions were effectively ephemeral, since they were only presented orally or because the graded submission was returned to the student. Now, submissions are archived electronically and are thus available for consultation, comparison, and various types of analyses previously impossible.

The paper-based system also allowed students to get points for assignments they had not actually completed, whereas they now have to submit written solutions for the assignments electronically before the classroom session. For programming assignments, students are now required to submit working programs through ECAutoAssessmentBox. While it would have theoretically been possible to enforce these requirement in the paper-based system, it would have resulted in an unmanageable workload for instructors.

On one hand, the demands for students' solutions are now much more explicit and rigid with respect to correctness, quality, and clarity. On the other hand, students can gain access to a larger number of alternative solutions and to typical error cases. Students also report 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.

For both programming and essay-like assignments, reviewing larger numbers of student submissions is now possible because the submissions are collected at a central 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.

Our motivation for automatic testing is twofold. On one hand, automatic testing reduces the workload of instructors, which on the other hand, allows instructors to assign more programming exercises to students, helping students to gain more programming practice. Our experience has shown that quite a number of students tries to avoid writing and testing programs and content themselves with non-working sketches. The automatic testing of programs enforces the requirement that programming submissions must be running programs.

Automatic testing is not intended 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. [3] 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.

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.” [2]. Gibbs and Simpson [5] 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.”

While the feedback provided by the automatic tests is very rudimentary (this is in part intentional, since they are not designed as a tutoring system), the immediate feedback is mentioned surprisingly often as very helpful by students in their responses to the questionnaire. 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. Thus, even though the automatic feedback may not yet be perfect, it represents a notable improvement for the students’ learning experience.

4 Conclusions

While computer support in the areas of instruction, feedback, and student tracking is not new, the eduComponents are distinguished by their component-based, document-oriented architecture on the basis of a general-purpose CMS. The eduComponents are extension modules for the Plone open-source CMS. Building on portable open-source software avoids license fees and vendor lock-in, and has benefits like complete control over the software and the data. Consequently, we have also released the eduComponents modules as open-source software[2], so that others can use them as well, and all users benefit from a larger community[3]. This ensures the continued maintenance of the eduComponents and increases the sustainability of eduComponents-based learning environments.

For users already familiar with Plone, the eduComponents are especially easy to use because the content types provided by the eduComponents (tests, assignments, submissions, etc.) behave like all other content types in Plone (texts, pictures, events, folders, etc.); for other users, the file-system-like organization of content makes it easy to learn. The modular approach of eduComponents, which makes it possible to start with just a single module, reduces the entry hurdle for the deployment of e-learning. Since Plone is a general-purpose CMS, it can be used to manage any Web content, thus offering uniform usage and administration for all types of content. The CMS also serves as *item bank*, a central repository

[2]Available from `http://wwwai.cs.uni-magdeburg.de/software/`

[3]For other users see `http://www.uni-magdeburg.de/PM_162_2006-highlight-162.html`

of tests, assignments, and solutions, enabling the reuse of teaching and learning materials.

The digital storage of assignments and solutions opens up many new learning and teaching possibilities. Online access to solutions helps instructors in detecting whether students have problems with an assignment; instant availability of complete solutions during face-to-face courses results in better motivation and stimulates discussion. Other possible uses include the distribution of anonymized submissions for peer review by other students. In computer science education, Zeller [9] and Sitthiworachart and Joy [8] report noticeable benefits for the learning of programming, and we are currently developing a peer review module for the eduComponents.

The ongoing evaluation of our eduComponents-based learning environment shows broad acceptence of the new system and the new procedures; in fact, most of our students have indicated that they would like to see it used in more courses.

References

1. M. Amelung, M. Piotrowski, and D. 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, New York, 2006. ACM Press.
2. A. W. Chickering and Z. F. Gamson. Seven principles for good practice in undergraduate education. *AAHE Bulletin*, 39(7):3–7, March 1987.
3. C. Douce, D. Livingstone, and J. Orwell. Automatic test-based assessment of programming: A review. *J. Educ. Resour. Comput.*, 5(3):4, 2005.
4. T. 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.
5. G. Gibbs and C. Simpson. Conditions under which assessment supports students' learning. *Learning and Teaching in Higher Education*, 5(1), 2004.
6. M. Piotrowski and D. Rösner. Integration von E-Assessment und Content-Management. In J. M. Haake, U. Lucke, and D. Tavangarian, editors, *DeLFI2005: 3. Deutsche e-Learning Fachtagung Informatik der Gesellschaft für Informatik e. V.*, volume P-66 of *Lecture Notes in Informatics (LNI)*, pages 129–140, Bonn, 2005. GI-Verlag.
7. D. Rösner, M. Amelung, and M. Piotrowski. E-Learning-Komponenten zur Intensivierung der Übungen in der Informatik-Lehre – ein Erfahrungsbericht. In P. Forbrig, G. Siegel, and M. Schneider, editors, *2. GI-Fachtagung Hochschuldidaktik der Informatik*, volume P-100 of *Lecture Notes in Informatics (LNI) – Proceedings*, pages 89–102, Bonn, 2006. GI-Verlag.
8. J. Sitthiworachart and M. 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.
9. A. 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.