E-Assessment as a Service

Mario Amelung, Katrin Krieger, and Dietmar Rösner

Abstract—Assessment is an essential element in learning processes. It is therefore not unsurprising that almost all learning management systems (LMSs) offer support for assessment, e.g., for the creation, execution, and evaluation of multiple choice tests. We have designed and implemented generic support for assessment that is based on assignments that students submit as electronic documents. In addition to assignments that are graded by teachers, we also support assignments that can be automatically tested and evaluated, e.g., assignments in programming languages, or other formal notations. In this paper, we report about the design and implementation of a service-oriented approach for automatic assessment of programming assignments. The most relevant aspects of our "assessment as a service" solution are that on the one hand the advantages of automatic assessment can be used with a multitude of programming languages, as well as other formal notations (as so-called backends); on the other hand, the features of these types of assessment can be easily interfaced with different existing learning management systems (as so called frontends). We also report about the practical use of the implemented software components at our university and other educational institutions.

Index Terms—Computer science education, programming, learning systems, learning control systems, modular computer systems, service-oriented architecture, web services, e-learning, computer-aided assessment, e-assessment, eduComponents.

1 Introduction

1.1 Motivation

Teaching and learning in a computer science curriculum are demanding tasks. This is due to the intellectual and scientific content, as well as to the institutional and organizational context. The latter may be characterized as follows:

- large numbers of students, especially in introductory courses,
- broad diversity between students with respect to, e.g., prior knowledge, working habits, and intellectual capacities, and
- a high drop out rate and a gap between the demands for skilled computer scientists and the number of successful students.

Exercises and/or laboratory practice are essential for the learning effect, since they provide opportunities for students to solidify the knowledge acquired in lectures and to apply their theoretical knowledge to practical problems. In its traditional format, exercise groups are centered around work on paper and a shared presentation medium, e.g., in its simplest guise, a chalkboard. However, we were dissatisfied with some aspects of this traditional way of teaching, practicing, and assessing which may be sketched as follows:

Before classroom sessions

- the teacher designs or chooses assignments for a weekly exercise sheet according to the state of the course,
- the exercise sheet may be distributed as a printed document or made available online, e.g, as a PDF document,
- students work through the exercise sheet at home.

During classroom sessions

- students present their solutions at the blackboard,
- tutor and peers give (spontaneous) feedback,
- peers take notes from the presentation,
- the tutor may take notes about student's performance.

As a variation, written submissions may be demanded for marking and grading by tutors. But there is always a delay between the submission and the reception of comments and/or a corrected version. For large groups of students, manual correction is labor- and time-intensive.

The problems are 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 a few programming assignments can be handed out. This situation is also unsatisfactory for students, because their solutions and their problems frequently could not be discussed in detail due to time constraints.

Since we are using the web-based content management system (CMS) *Plone*[1] to deliver learning material (e.g., slides, notes, or reading lists) to our students, it appeared obvious to employ this CMS as a portal for the management of assignments, tests, and submissions. Using a CMS as the basis for managing students' assignments in the form of electronic documents is in many ways advantageous compared to traditional paper-based assignments. A CMS makes it much easier to handle, assess, store, and reuse assignments, and it allows new learning arrangements that are hardly possible without such a technological basis (e.g., hall of fame for students' submissions or student peer reviews).

- *M. Amelung is with Eudemonia Solutions AG, Sandtorstr. 23, 39106 Magdeburg, Germany. E-mail: amelung@eudemonia-solutions.de.*
- *K. Krieger and D. Rösner are with the Fakultät für Informatik/IWS, Otto-von-Guericke-Universität, Universitätsplatz 2, 39106 Magdeburg, Germany. E-mail: {kkrieger, roesner}@ovgu.de.*

Manuscript received 3 Mar. 2010; revised 2 June 2010; accepted 2 Aug. 2010; published online 17 Aug. 2010.
For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLT-2010-03-0021.
Digital Object Identifier no. 10.1109/TLT.2010.24.

1. http://plone.org/.

The benefits for the learning processes are even more substantial when the electronic submission of students' assignments is coupled with automatic assessment (i.e., automatic testing, marking, and grading). Automatic assessment allows timely, almost immediate feedback for students, which is known to be an additional motivating factor (cf. [3]).

The decision for using a CMS as an LMS over using a "native" LMS has historic, as well as practical reasons. The workgroup's website has been based on Plone before e-learning or blended learning strategies have been used in teaching. With increasing interest in e-learning and in the ways in which e-learning technologies can be integrated into existing structures and technologies (organizational, as well as technical), the idea of enhancing Plone with additional components, in order to convert Plone into an LMS, arose. As we were unwilling to administer and maintain a second native LMS, which provides a good portion of the same functions as a CMS, we have designed, implemented, and deployed a number of modules for Plone which extend the CMS with specific e-learning functionality (cf. [4] and [5]). These modules—collectively called eduComponents[2]—provide specialized content types offering the following main functions (see also [6] and [7]):

- ECLecture is a product for managing lectures, seminars, and other courses. It also serves as "portal" to all course-related materials and handles registration for courses.
- ECQuiz supports the creation and delivery of interactive multiple-choice tests. It can be used for formative tests to quickly assess the performance of a class without the need for extra grading work.
- ECAssignmentBox allows the creation, submission, and grading of essay-like assignments. The assessment process is semiautomated, 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.

The eduComponents modules can be used separately, or in combination and, since many basic functions are already provided by the CMS, they implement much of the standard functionality required in an e-learning environment. Deploying the eduComponents turns Plone into a fullfledged, tailormade LMS.

However, we wished to offer our students more timely feedback and more detailed discussion on their programming assignments too. 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. Therefore, students should be given frequent programming assignments, but assessing a large number of such assignments is a time-consuming and labor-intensive task.

For this reason, we have been targeting a system for assessing assignments in computer science education which provides automatic testing of programming assignments *as a service*, and can therefore be flexibly integrated into existing e-learning environments.

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

1.2 State of the Art

The term *assessment* is often used to summarize all activities that teachers use to help learners learn and to quantify the learning progress and outcomes (cf. [8]). The latter, in particular, means that assessment measures and documents the knowledge, skills, and attitudes of an individual learner, a learning community (e.g., class, course, or workshop), or an educational institution.

In computer science education—especially in introductory programming courses—a significant portion of the coursework consists of programming assignments that need to be assessed. Since the submitted assignments should be executable programs with a formal structure, the obvious thing to do would be to automatically evaluate these programs using compilers and interpreters, or specialized frameworks for static and/or dynamic testing. Common advantages of automatic assessment tools for computer programs are speed, availability, consistency, and objectivity of assessment. However, automatic tools emphasize the need for careful pedagogical design of the assignment and assessment settings. To effectively share assessment solutions already developed, better interoperability and portability of these tools is highly desirable (cf. [9]).

The first systems supporting marking and grading of student solutions for programming exercises were developed and used as early as 1960 (cf. [10]). Since that time, the motivation (among others, large numbers of students) and the same topics remained relevant: Some of them are security, plagiarism detection, and automatic assessment.

Nowadays, there is a multitude of—mostly web-based—systems for automatic testing of programming assignments which are used to supplement teaching in computer science. Some of these systems are specialized in a specific programming language or test method, e.g., *TRAKLA2* [11] for algorithm simulation exercises, *Scheme-robo* [12] for programming assignments in Scheme, *AutoGrader* [13] for Java programs, or *JACK* [14], as well for programs written in Java.

In addition, other systems like *CodeLab*[3] (Java, C/C++, and Python programs) or Addison Wesley's portal *MyCodeMate*[4] (Java and C/C++) support several programming languages. And there are also systems that support any programming language and any test method since the real testing functionality is encapsulated in modules, e.g., *CourseMarker* [15], *BOSS* [16], or the AT(x) framework [17].

The project *Praktomat* [18] from the University Passau is devoted to better quality control of programming assignments. It offers—additionally to compiling and testing of program code—the possibility of checking assignments for their conformity to the *Java Code Conventions*. Other projects like *WeBWorK* [19] or *Web-CAT* [20] focus on learning about test-driven software development. Systems like *DUESIE* [21] even enable the computer-assisted analysis of UML assignments.

The *autotool* system [22] from the University of Leipzig accepts students' submissions to assignments in theoretical computer science and supports exercises on grammars, regular expressions, automata, or graph properties.

3. http://www.turingscraft.com.
4. http://www.mycodemate.com.

Almost all of these systems have the common property of providing (along with the actual testing of programming assignments) functionality for managing users, courses, assignments, and submissions. This results in a strong coupling of testing and grading functionality with these course management functions. Therefore, the transfer and integration of the automatic testing into existing LMSs is not easily realized. Thus, the usage of these systems inevitably leads to administering two or more systems and keeping redundant data. Furthermore, those systems are difficult to extend and to adapt to one's own requirements. They are built for the purpose of testing programs in a certain language or employ a certain test method. This results in rather inflexible, monolithic systems, not created for possible extension by additional functionality.

Another group of such systems are mainly LMSs that work in a vice versa manner compared to the systems mentioned before. LMSs like *Moodle*,[5] *Blackboard*,[6] or *OLAT*[7] are mainly designed and implemented for the management of courses, classes, grading, and learning materials. They also offer tools for assessment of students, including questionnaires, single/multiple-choice tests, file uploads (e.g., PDF files or voice files), or free text answers. Tools for automatic marking and grading, however, are rare for those systems, especially for automatic testing (and marking and grading, respectively) of programming assignments.

We found two plugins that extend Moodle by functionality for automatic testing of programming assignments. The project *Epaile* was initiated during Google's "Summer of Code 2007" and "has the objective to develop a plugin for moodle, making it able to grade computer programming assignments automatically" [23]. However, the status of this project remains unclear, since there doesn't seem to exist any release of this Moodle plugin. The project *OnlineJudge*[8] is a special assignment type for the Moodle LMS. This plugin can automatically grade programming assignments by deploying test cases customized by the instructor. It supports the testing of assignments in several languages, including C/C++, Java, Python, and others. The tests can be run on the server machine running Moodle itself (only applicable for C/C++ on Linux) or via *ideone.com* (an external online compiler and debugging tool).

However, OnlineJudge also ties the testing functionality of programming assignments strongly to the used LMS, which is Moodle in this case, making it impossible to use OnlineJudge in conjunction with other learning management systems and, in particular, with our Plone-based learning environment.

We could not find any such testing functionality for Blackboard or WebCT, respectively.

In order to characterize and compare different systems for automatic assessment in a more structured way, we work with the following criteria that will as well be used in the synopsis of selected assessment systems[9] below (see Fig. 1):

5. http://moodle.org.
6. http://www.blackboard.com/.
7. http://olat.org.
8. http://code.google.com/p/sunner-projects/wiki/OnlineJudgeAssignmentType.
9. We selected examples from each group of systems mentioned in the text before.

System/Project	TRAKLA 2	Scheme -robo	Auto-Grader	Code-Lab	My-Code-Mate	Course-Marker	BOSS	AT(x)	Moodle	Black-board
Criteria										
Assessment type										
‣ objective	☐	☐	☐	☐	☐[6]	☐	☐	☐	✓	✓
‣ subjective	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Automatic assessment										
‣ programming assignments	☐	✓[1]	✓[1]	✓[2]	✓[2]	✓[3]	✓[3]	✓[3]	✓[2]	☐
‣ other formal notations	✓	☐	☐	☐	☐	✓	☐[6]	☐	☐	☐
‣ different test methods	☐	☐	☐	☐[6]	☐[6]	✓	✓	✓	☐	☐
Integration with other learning platforms	☐	✓	☐	☐	☐	☐	☐	✓[4]	☐	☐
Extensibility	☐	☐	☐	☐	☐	☐	☐	☐	✓	☐
Open source	✓	✓	☐	☐	☐	☐	☐	☐	✓	☐
Support[5]	✓	☐	☐	✓	✓	✓	✓	☐	✓	✓

[1] single programming language
[2] two or more programming languages
[3] (in principle) any programming language
[4] submissions via WebAssign
[5] latest release is not older than two years
[6] unknown

Fig. 1. Synopsis of systems for automatic assessment.

- Which assessment types are supported: objective assessment only (e.g., multiple choice, true/false, matching, and completion items) or subjective assessment (e.g., extended-response questions, essays, or programming assignments) as well?
- For automatic assessment: Is the system designed for a single programming language or is it usable (in principle) with any programming language? Is the scope even broader and includes as well automatic assessment of formal systems beyond traditional programming languages?
- Is the system confined to a single test method or does it support a variety of test methods?
- Is extensibility of the system by users explicitly foreseen and supported?
- Is the system available as open source or is it closed source only?
- Is the assessment system standalone only or is integration with other systems supported, in particular, with other learning management systems?
- Is the system still supported and further developed (i.e., is the latest release not older than two years)?

1.3 Requirements

Based on our motivation (see Section 1.1) and the actual needs in teaching at our institution, an e-assessment system for testing programming assignments has to satisfy the following requirements:

- flexible integration of test and grading functionalities in existing learning environments without redundant user management and data storage,
- automatic evaluation of programming assignments in different programming languages with different test methods,
- automatic evaluation of assignments in other formal notations, e.g., regular expressions, XSL transformations (XSLT), and UIMA analysis engines,[10]
- assessment and grading of assignments that require short text answers in natural language,
- easy extension to provide additional assignment types, programming languages, and test methods.

10. http://incubator.apache.org/uima/.

From the systems mentioned in Section 1.2, none met all of our requirements (cf. Fig. 1). Especially, the lack of flexible integration into existing LMSs was grave, since we were using Plone for our workgroup's website, and since Plone already offers a lot of functionality that is useful in organizing teaching and learning. Thus, we decided to look for possiblilities to integrate automatic assessment functionality (in particular, for programming assignments) into a heterogeneous learning platform.

1.4 Structure of This Paper

This paper is organized as follows: First, we will introduce a novel service-oriented approach for the development and deployment of flexible and reusable software components for automatic assessment. To demonstrate our approach, we afterward go into detail explaining the necessary steps that have to be taken in order to specify new services for testing programming assignments. In Section 3, we elaborate on the development of frontend components for user interaction with the testing system. We will show what specific frontends and backends have already been developed and give a short preview of additional components we plan to implement. Finally, we report about our experiences with the deployment of the introduced components and reflect on the effects on teachers and students. We will also give an outlook on future development of our e-assessment and e-learning approach.

2 A Service-Oriented Approach

2.1 Design Decisions

In contrast to the systems introduced in Section 1.2, our approach focuses on a clear separation of all aspects regarding the management of learners, assignments, and submissions from the actual testing of programming assignments.

To achieve this goal, we employ a *service-oriented architecture* (SOA). An SOA is a framework for the integration of (business) processes as secure, standardized components—-so-called services—that can be reused and combined to meet varying requirements (cf. [24]). A service is a software component whose functionality is offered platform independently through an interface over the network. Service orientation requires loose coupling of services, which communicate with their corresponding consumers by passing data in a well-defined, shared format, or by coordinating an activity between two or more services (cf. [25]).

The actual testing of programming assignments is highly dependent on the kind of test method, programming language, or other formal notation involved. For example, programs can be evaluated by using static and/or dynamic tests. For the latter, the output of a program can be compared to that of a model solution, or the assignment can be tested for properties which must be fulfilled by correct programs. Hence, all aspects regarding the exact testing should be encapsulated and implemented in self-contained services—we call them *backends*. Backends are functional building blocks of our SOA, which provide test and assessment facilities over standard Internet protocols independent of platforms and programming languages.

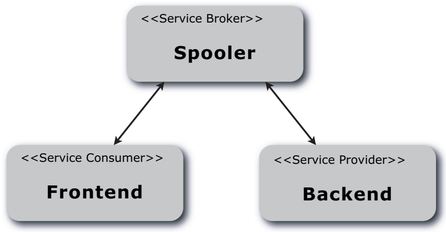


Fig. 2. Roles in a service-oriented architecture and its equivalents in our approach.

Teaching and learning are core business processes within educational institutions. These processes are typically supported by an LMS. Following the above-mentioned idea of separating all concerns related to managing from testing and assessment, learning management systems play the role of service consumers in our SOA approach. In the following, we will use the term *frontend* for the LMS employed. Common functions of a frontend are, for instance, storage of assignments and solutions, proper treatment of submission periods and resubmissions, communication of results to students, or statistics for individual students and whole cohorts. For automatic testing purposes, frontends access the functionality provided by the backends.

To enable uniform access to the backends and a preferably loose coupling of frontends and backends (avoiding too many point-to-point connections), we introduced a third component, the so-called *spooler*. Similar to a printer spooler, it manages a submission queue, as well as a variety of backends, and provides the following functions:

- add new submissions for testing,
- get results from tests performed by a backend,
- show status information (e.g., available backends and number of submissions in queue),
- add or remove backends,
- get required input fields for testing with a certain backend,
- get available test method options.

In this manner, the spooler plays at first the role of a service broker in our SOA, but it is also a service provider for different frontends, as well as a service consumer, since it uses varying backends (cf. Fig. 2).

Implementing the spooler and backends in a service-based way results in a high degree of interoperability and flexibility and also offers the option to combine a multitude of frontends and backends. It also ensures the integration of any backend—even in heterogeneous system environments. The encapsulated testing functionality in the backends can be reused and extended.

Fig. 3 shows the three key components of the service-oriented approach and examples of their potential realizations. Having already discussed the core functionality of the spooler, we will go into detail about the specification and implementation of frontends and backends in the following sections.

2.2 Frontend Specification

The basic idea is that instructors create electronic/online *assignment boxes*, into which students submit their answers or solutions. These submissions are stored as *assignments*

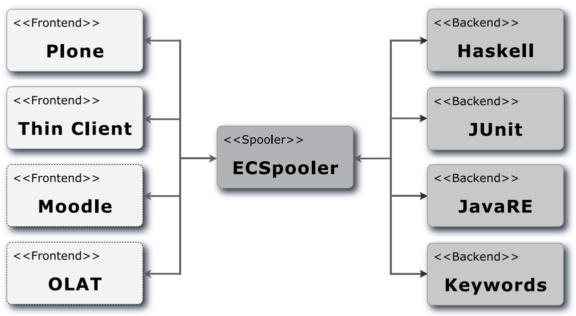


Fig. 3. Components and realizations of our service-oriented approach for e-assessment.

inside the assignment box. Each submission, as well as all necessary test data are sent to the spooler, which, in turn, passes it on to the selected backend. The assessment of student submissions starts with the submission of a student's answer and ends with the grading of an assignment by the instructor. This process can be modeled as a workflow, i.e., from the initial submission to the final grading, submissions are put through a number of workflow states.

Besides standard attributes (e.g., title, author, and creation date), assignment box objects have a number of attributes to realize specific functionality:

- assignment text,
- answer template,
- submission period, and
- number of attempts.

When an instructor creates a new assignment box, it has to be associated with a certain backend. Therefore, the box has to communicate directly with the spooler (see Section 2.1) which returns a list of available backends and their input fields that are needed for testing.

Those input fields may vary according to the chosen backend and the assignment box has to dynamically create the user interface, i.e., for a test-case-based backend, the instructor has then to type in test data and a model solution, whereas he has to specify unit tests for respective unit test backends.

Students can read the assignment text and submit their answers. If the submission period is restricted, submissions are only allowed until the submission period has ended. Multiple attempts to answer assignments are allowed—up to the maximum number of attempts specified by the instructor. Fig. 5 shows roles and actions that must be implemented by a frontend.

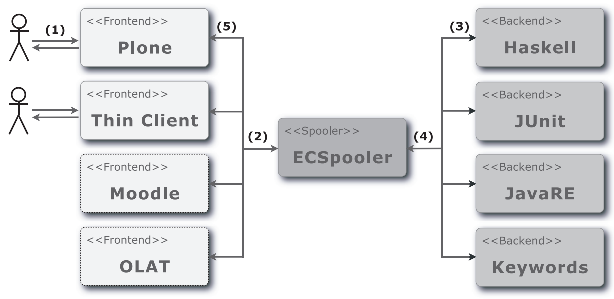


Fig. 4. Processing a student's submission for automatic testing.

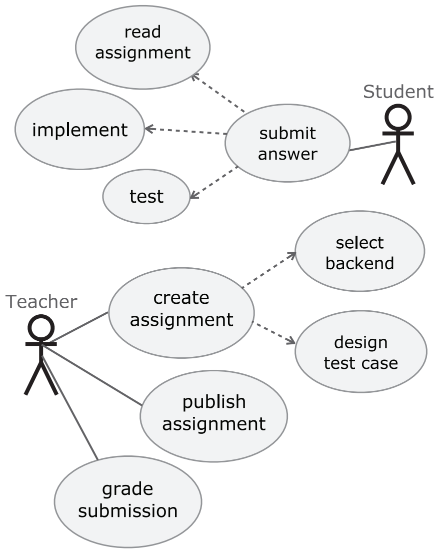


Fig. 5. Roles involved and actions provided by a frontend.

Processing a submission is shown in Fig. 4:

1. The learner types a solution (program code) into the frontend,
2. The frontend sends the submission to the spooler and gives information about which backend to use,
3. The spooler forwards the submission to the appropriate backend. The backend executes the compiler and/or interpreter with the submitted code,
4. The backend returns status messages and possible error descriptions to the spooler, and
5. The frontend polls the spooler for the result of the test, fetches it, and displays it as feedback to the user.

Later on, in Section 3, we will describe two existing frontends in more detail and show how this specification was realized.

2.3 Backend Specification

In the following, we will illustrate which influencing factors have to be taken into account for the development of new backends.

2.3.1 (Programming) Language and Test Method

For the development of new backends, the first step is to answer two essential questions:

1. *What* is to be tested? and
2. *How* should the tests be pursued?

To answer the first question, it must be determined which *programming language* or other formal notation should be supported by the backend. This decision will affect all subsequent steps.

The answer to the question, how a submission should be tested, is given by the decision for a certain *test method* (e.g., static versus dynamic tests; cf. [26]). This decision is, however, dependent on the chosen programming language. If, for example, submissions should be tested dynamically with unit tests, this requires the availability of a unit test framework for the chosen language. Furthermore, the conditions that lead to an early termination of the test run have to be determined. For instance, dynamic tests need not be run if a syntactical test has already failed. It could also be defined that a test run (comprising a number of tests) should be terminated as soon as a single test has failed.

2.3.2 Input and Output

The chosen test method determines requirements for the *input data*, i.e, information that has to be given by the instructor and information that is enclosed in the actual submission. In most cases, a student's submission contains the answer to an assignment in one or more text or source code files.

The instructor has to provide information about the constraints of the assignment. For enabling automatic testing of the submission—in addition to the description of the problem to solve—the learner has to have the following information:

- name of the functions he has to implement,
- number and types of function arguments,
- type of the return value.

The instructor also has to provide the information necessary for testing. For a comparison with a model solution, it is necessary to have such a model solution and also test data (input data). In this case, the instructor also has to set whether the results of the submission should be identical to those of the model solution or if in case of a list-valued result, permutations are allowed as well.

In addition to input data, the *output* of the backend must also be defined exactly since this value will be forwarded as feedback to the learner. If all tests have been passed successfully, this should result in a positive feedback. In the case of errors, there has to be a detailed feedback about the type of error such that the student gets hints about the problems with his solution. Hence, messages from the compiler or interpreter about syntactical or runtime errors should be forwarded to the learner, as well as information about failed tests, test data, and expected results. It must be pointed out that the available information that could be used as feedback for the user depends upon the chosen programming language and test method.

2.3.3 Compiler and Interpreter

The choice of a particular compiler and/or interpreter may yield certain preconditions and limitations. In particular, the type and information content of return values of a compiler or interpreter have to be analyzed with respect to its use as feedback from the backends. Those return values could be about successful compilations or error messages.

2.3.4 Security

During dynamic tests unknown and potentially faulty source code will be executed. Thus, certain security aspects have to be considered. Without precautions, a submission can execute—with the privileges of the backend user—any functions and programs, run denial-of-service attacks, or spy on information about other users, the system, or assignments (especially the model solution).

This results in security requirements that have to be taken into consideration in later deployment of the backends, depending on programming language and platform. For instance, a restricted interpreter or a sandbox environment can be used for program execution. Other possibilities include the deployment of additional software to limit the access to system calls, e.g., *Systrace* [27].

Furthermore, it is reasonable to set a time limit for testing the submitted programming code. If this time limit is exceeded, the execution of the current submission is aborted because the code is suspected of containing infinite recursions or infinite loops.

3 Frontend Implementation

The service-oriented approach and the loose coupling of software components, respectively, allows the use of the spooler and its registered backends with literally a multitude of frontends. In this chapter, we introduce two of our already existing frontends—ECAutoAssessmentBox and a lightweight Java frontend—and give an outlook on the undergoing development of two additional backends that will enable Moodle and OLAT to use the testing functionality of ECSpooler.

3.1 ECAutoAssessmentBox

As mentioned in Section 1.1, our learning environment is based on Plone and the eduComponents. The module ECAutoAssessmentBox from the eduComponents offers instructors facilities to create online assignments and to accept submissions for automatic testing with a number of different backends. As an extension module for Plone, ECAutoAssessmentBox is implemented in Python. To communicate with ECSpooler and its backends, *Python's XML-RPC client API* is used. XML-RPC[11] is a remote procedure call method using HTTP as the transport protocol and XML for encoding. It allows complex data structures to be transmitted, processed, and returned. With it, a client can call methods on a remote server.

Fig. 6 shows, as an example, the options and input fields for the JUnit backend generated by ECAutoAssessmentBox. This backend runs the student solution on a set of unit tests.

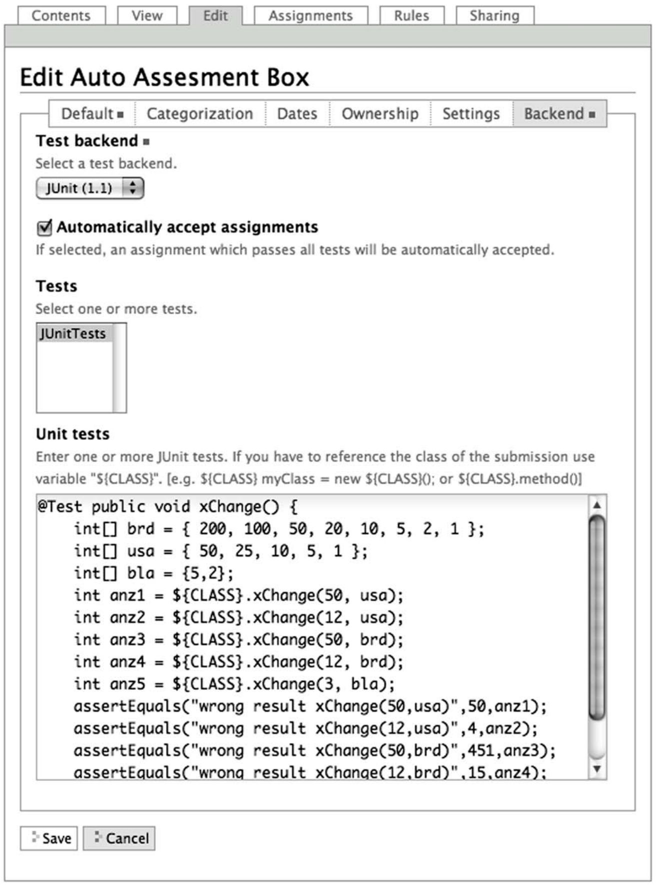
Students submit their answers via the web interface of ECAutoAssessmentBox. They can read the assignment text and submit their solution either by typing it into a textbox or via file upload. If the submission period is restricted, information about the deadline will also be displayed.

The result of a test run will be shown to the learner immediately and the submission will be marked either "passed all tests" or "failed." If a submission fails, then the given feedback includes the test case and the expected result (for an example, see Fig. 8).

3.2 Stand-Alone Thin Client

Besides ECAutoAssessmentBox, other arbitrary frontends can be used in conjunction with ECSpooler and backends. For proof of concept, we implemented a lightweight, stand-alone client (see Fig. 7) written in Java which communicates with a web service based on SOAP,[12] which, in turn, communicates with ECSpooler.

11. http://www.xmlrpc.com/.
12. Simple Object Access Protocol (SOAP) is a standard for exchanging structured information in the implementation of web services (cf. http://www.w3.org/TR/soap/).



Edit Auto Assesment Box

Test backend
Select a test backend.
JUnit (1.1)

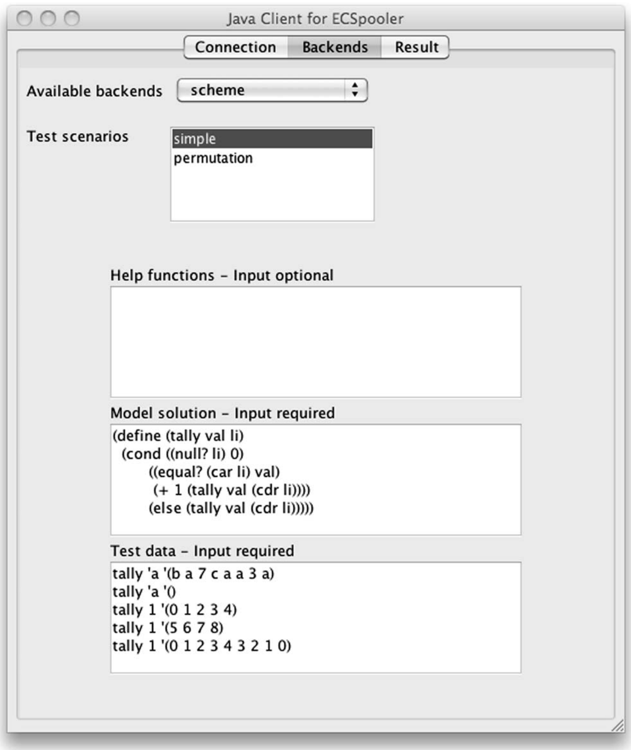
☑ **Automatically accept assignments**
If selected, an assignment which passes all tests will be automatically accepted.

Tests
Select one or more tests.
JUnitTests

Unit tests
Enter one or more JUnit tests. If you have to reference the class of the submission use variable "${CLASS}". [e.g. ${CLASS} myClass = new ${CLASS}(); or ${CLASS}.method()]

```
@Test public void xChange() {
    int[] brd = { 200, 100, 50, 20, 10, 5, 2, 1 };
    int[] usa = { 50, 25, 10, 5, 1 };
    int[] bla = {5,2};
    int anz1 = ${CLASS}.xChange(50, usa);
    int anz2 = ${CLASS}.xChange(12, usa);
    int anz3 = ${CLASS}.xChange(50, brd);
    int anz4 = ${CLASS}.xChange(12, brd);
    int anz5 = ${CLASS}.xChange(3, bla);
    assertEquals("wrong result xChange(50,usa)",50,anz1);
    assertEquals("wrong result xChange(12,usa)",4,anz2);
    assertEquals("wrong result xChange(50,brd)",451,anz3);
    assertEquals("wrong result xChange(12,brd)",15,anz4);
```

Fig. 6. Web interface generated by ECAutoAssessmentBox for a programming assignment in Java (JUnit backend).



Available backends: scheme

Test scenarios: simple, permutation

Help functions – Input optional

Model solution – Input required

```
(define (tally val li)
  (cond ((null? li) 0)
        ((equal? (car li) val)
         (+ 1 (tally val (cdr li))))
        (else (tally val (cdr li)))))
```

Test data – Input required

```
tally 'a '(b a 7 c a a 3 a)
tally 'a '()
tally 1 '(0 1 2 3 4)
tally 1 '(5 6 7 8)
tally 1 '(0 1 2 3 4 3 2 1 0)
```

Fig. 7. User interface of a frontend implemented as thin client in Java.

Thus, the Java client can be used by instructors to quickly design and test their assignments with different backends, without the need for a full installation of an LMS.

3.3 Moodle and OLAT

In the near future, we intend to implement frontends for Moodle and OLAT, so that the testing functionality of ECSpooler and the backends can be used also with these two LMS. Developing and implementing frontends for Moodle and OLAT means that additional content types will be created. These content types will be based on the LMS's own existing assignment types (e.g., free-text assignments), exploiting the already existing functions, such as managing users, groups, assignments, deadlines, or number of attempts.

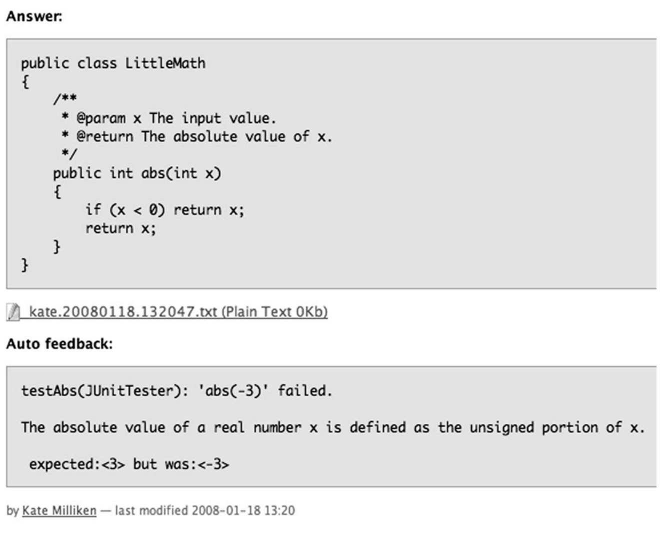
4 Backend Implementation

In recent years, we have developed and deployed backends for XML, as well as for the programming languages Haskell, Scheme, Erlang, Prolog, Python, and Java. However, with the appropriate backends, submissions in other formal notations can also be tested and even natural language assignments can be analyzed.[13]

All backends are implemented as web services using *Python's XML-RPC server API*. A so-called input schema is used to describe all input fields that are necessary for a complete specification of a test run. For example, for a test-data-based analysis, a model solution and a couple of function calls have to be provided. Furthermore, the schema defines at least one so-called *test method option*. Those test method options allow instructors to choose between different compilers or interpreters for a programming language or different comparison methods (e.g., exact match versus tolerance match for floating-point values).

Backends are derived from general backend classes. These classes provide a number of standard functions like starting and stopping of backends, or registering a backend to a spooler. Thus, the development of new backends is

13. We have already experimented with style checking and keyword spotting (cf. [28]).


Answer:

```
public class LittleMath
{
    /**
     * @param x The input value.
     * @return The absolute value of x.
     */
    public int abs(int x)
    {
        if (x < 0) return x;
        return x;
    }
}
```

kate.20080118.132047.txt (Plain Text 0Kb)

Auto feedback:

```
testAbs(JUnitTester): 'abs(-3)' failed.

The absolute value of a real number x is defined as the unsigned portion of x.

 expected:<3> but was:<-3>
```

by Kate Milliken — last modified 2008-01-18 13:20

Fig. 8. Example feedback from the JUnit backend.

reduced to the definition of input and output schemas and methods for the execution of the concrete tests.

4.1 JUnit

In this section below, we will briefly and step-by-step demonstrate how a new backend can be specified (cf. Section 2.3) using the example of the *JUnit backend*.

4.1.1 Programming Language and Test Method

The JUnit backend is one possible option when learners implement solutions to programming assignments in the Java programming language.

First, solutions have to be syntactically correct. The logical correctness in the sense of proper working of the program shall be automatically evaluated with the help of appropriate unit tests. A unit test is "a test that exercises a relatively small executable. In object-oriented programming, an object of a class is the smallest executable unit, but test messages must be sent to a method, so we can speak of method scope testing. A test unit may be a class, several related classes (a cluster), or an executable binary file. Typically, it is a cluster of independent classes" [29].

The utilized framework is *JUnit*.[14] A solution is considered correct when all tests have been passed without errors. If one test fails, the complete test run will be aborted.

4.1.2 Input and Output

The instructor provides test data in the form of unit tests. In addition, imports and helper functions can be provided. The signature of the method (name, type of the arguments, and return value), that is going to be called, is defined in the assignment text. The learner submits his solution as source code.

The feedback for the learner contains information about the syntactical correctness and whether *all* unit tests have been passed without errors. As soon as one test fails, a negative feedback will be returned. This feedback contains messages from the Java compiler in the case of syntactic errors and the return value of the according JUnit test in the case of logical flaws. In the case of such errors, the anticipated and actually gained results will be presented to the learner (see Fig. 8 for an example). Thus, he gets the opportunity to recognize mistakes in his solution and revise it accordingly.

4.1.3 Compiler and Interpreter

The Java compiler is used to analyze the syntax in order to detect syntactic errors. If no errors are found, an executable program file will be created which is then used to run the dynamic test itself. The provided unit tests are executed by the Java interpreter and applied to the compiled program. Potential error messages will be collected and serve as direct feedback for the learner.

4.1.4 Security

First of all, security is granted by the backend being executed as an unprivileged user on a separate server host. In addition, certain function calls are disabled by using *Systrace* and the execution of the submission will be aborted if a certain time limit is exceeded.

14. http://www.junit.org/.

4.2 Haskell

The *Haskell backend* takes advantage of the fact that in pure functional programming languages, programs are functions whose values can be compared directly. The equality criterion can be specified by the teacher on a per task basis:

- direct match: output of student and model solution must be equal for the same test data,
- for lists as results: accept lists with same elements but in different order as equal (i.e., identical modulo permutation),
- for numerical algorithms: define a tolerance to account for the effects of rounding.

Using the Haskell backend, teachers have to select at least one criterion for equality and they have to specify a model solution and test data.

If a single test case fails, the whole evaluation stops immediatly and the backend returns the corresponding function call, as well as the expected and actually received result.

Another backend for Haskell uses QuickCheck [30] for testing Haskell programs automatically. Teachers have to define formal specifications in the form of properties which a correct solution should satisfy. This will be tested with a large number of randomly generated test data. Feedback is given in the form that either all test data fulfilled all properties or which property failed on which data.

5 Practical Use

In the following, we discuss experiences from three applications of ECAutoAssessmentBox, ECSpooler, and backends for automatic assessment in computer science education, two of them at a location and within a group that is completely independent from the developers group.

5.1 Use Case 1: University Magdeburg

In Magdeburg, we are using the eduComponents modules as part of a blended learning strategy which consists of lectures, electronic exercise work, and exercise groups as regular classroom sessions.

The eduComponents have been actively used in all our lectures for several semesters. This includes introductory and advanced lectures in programming like "Algorithms and data structures," "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 and regular expressions.

Since October 2008,[15] we have published approximately 2,700 instances of ECAutoAssessmentBox and automatically evaluated about 30,000 submissions from students.

The most recent and broad scale usage in programming was from winter semester 2008/2009 to summer semester 2010 in the lecture "Algorithm and Data Structures" with three hours lecture and two hours exercise per week. This course is obligatory for all computer science bachelor

15. In October 2008, we migrated the eduComponents to Plone version 3.

students in their first semesters. It is essential that the students deepen their understanding by solving programming tasks. This can only be achieved when exercises and practice are very intensive.

An exercise group comprises approx. 15-20 students and is headed by a tutor. The tutor of the exercises—this is either an assistant or an advanced student—has prior access to all submissions. We therefore demand that students submit programming assignments several hours prior to the weekly group meeting and get them prechecked by ECAutoAssessmentBox (see Section 3.1). This facilitates better preparation for 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 the group session, all these documents are available online.

5.2 Use Case 2: University Rostock

At the University of Rostock, there exists a course called "Abstract Data Types" that is intended to introduce students to specification and Java programming. This course was taught for the first time in winter semester 2007/2008. Participants come from different course programs like computer science, business informatics, and information technology.

The course consists of lectures, exercises, and lab classes. Additionally, students have to perform specification and programming tasks at home. A minimum of 50 percent of the possible marks are necessary to get the admission to the exam.

The colleagues from Rostock have chosen to use the eduComponents (cf. [1]) because marking is a very time-consuming activity, and this is often the reason to reduce the number of problems given to the students. This contrasts with the need for significant amount of training via larger numbers of exercises during the first year.

The eduComponents—especially ECAutoAssessmentBox—proved to be very helpful to support this idea and to give students enough problems to train with. As assignments, students received algebraic specifications and the signature of the Haskell function as an answer template. They then had to implement the axioms as Haskell functions.

With the help of ECAutoAssessmentBox in conjunction with the Haskell backend, those submissions (i.e., specifications) could be tested for correctness according to a model solution.

The experiences with the system were very positive with both the Haskell and Java backend. Markers were allowed to have a look at the solutions in detail and to provide specific marks, which improved efficiency a lot. Soon after exposure to the system, tutors decided to use ECAutoAssessmentBox additionally for lab hours. This demonstrates how well the system has been accepted.

5.3 Use Case 3: LMU Munich

This third use case is even more interesting in the manner that the Institute for Computer Science at the Ludwig-Maximilians University Munich (LMU) has developed a backend for the programming language SML.[16] This backend is an adopted variant of the Haskell backend (cf. Section 4.2).

The Institute for Computer Science provides its own LMS for course management. This system, however, does not offer automatic assessment for programming assignments. Thus, an own installation of Plone with ECAutoAssessmentBox and ECSpooler has been used in conjunction with the developed SML backend for the lecture "Programming and Modeling" in summer term 2008. This lecture was attended by about 200 first-year bachelor students whereas 40 students used the automatic assessment system to prepare for the exercise classes.[17] During the semester, 43 assignments have been given to the students, and the system counted 1,047 submissions with automatic testing.

The effort for the development of the SML backend was relatively low. It took one day to derive the new SML backend from the Haskell backend. This illustrates that it is very easy to integrate test functionality for assignments in other programming languages.

6 Evaluation

In our experience, three factors are highly interrelated when issues of learning technology have to be discussed:

- didactical and pedagogical goals,
- issues of regulations for learning organization ("the rules of the game" or policy),
- opportunities and possibilities enabled by learning technology.

A lot has changed in our teaching since we started with the enterprise that later was termed eduComponents, and experimentation and innovation still goes on.

However, since we have no control groups that experience learning *without* eduComponents, a statistical comparison of learning outcomes with and without the system is not possible. We therefore use an evaluation procedure that takes the subjective judgements of the course participants into account.

At the end of each semester, we ask our students to complete a questionnaire on their experience with the eduComponents 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 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 classroom session. Students also report that they work more diligently on their assignments because the teachers can now access and review all submissions.

16. Standard ML, a functional programming language, based on Meta Langunage (ML).

17. Attending the excercise classes for this course were optional and not mandatory for admission to the final exam.

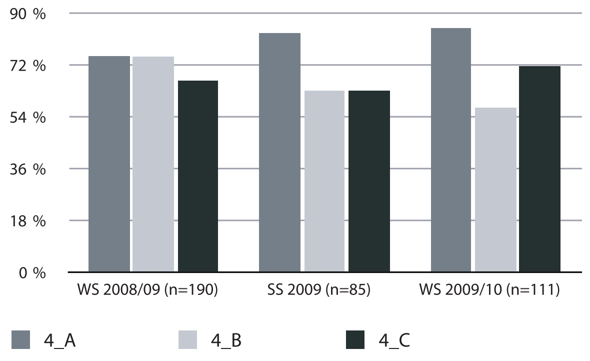


Fig. 9. Percentage of students that "fully agree" or "agree" with statements 4_A, 4_B, and 4_C on effects of automatic testing on the learning process.

Three questions especially deal with the effects of the automatic evaluation of programming assignments as implemented by ECAutoAssessmentBox, ECSpooler, and backends:

- 4_A "The automatic evaluation of programming assignments with immediate feedback is altogether very helpful."
- 4_B "The automatic evaluation motivates me to write running and correct programs."
- 4_C "The automatic feedback helps me to find errors in my programs."

As shown in Fig. 9, students gave consistently good ratings for the automatic assessment of their programs over the last three semesters.

Feedback from instructors at the universities Magdeburg and Rostock was collected through informal interviews. Instructors commented that the administration and review of student submissions for programming assignments was much easier with the eduComponents. They also reported that electronic submissions helped them in the preparation of classroom sessions and in the early detection of problems.

6.1 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 regarding 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.

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.

6.2 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 (cf. [31]). 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 again demand for policy decisions.

6.2.1 Unexpected Usage

ECAssignmentBox has been designed and implemented as a 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 does 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.

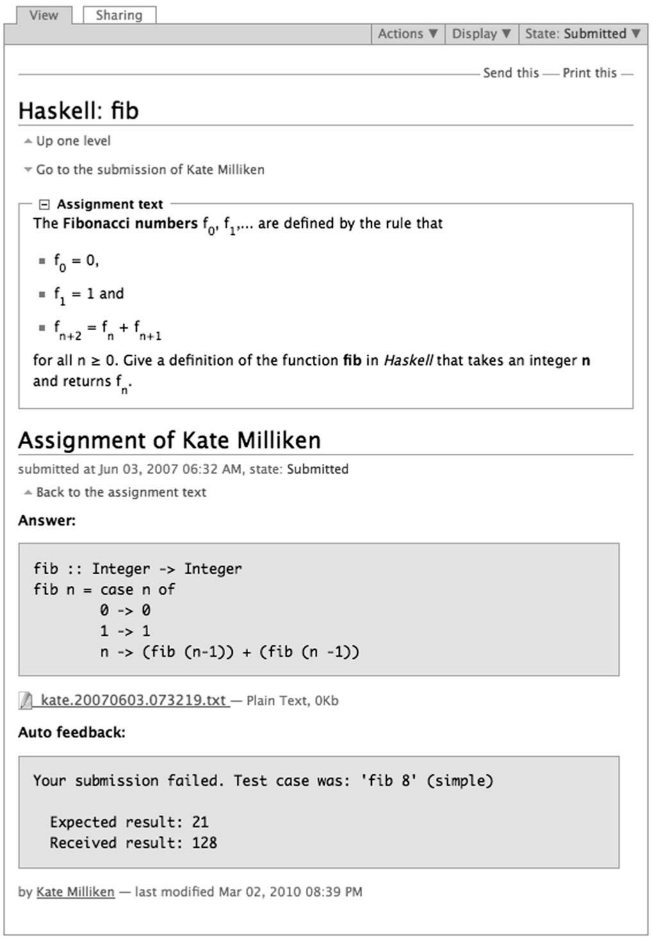
6.2.2 Unintended Usage

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 attempts. 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.

6.3 Reflection

Using a CMS as the basis for managing students' assignments in the form of electronic documents is in many ways advantageous compared to paper-only-based assignments. This may seem to be only a minor change, but we will illustrate how this move changes the organization of the learning processes and what new opportunities for learning result from this technology-enabled paradigm shift.

A CMS makes the handling, assessment, storing, and reuse of assignments much easier and it allows for new learning arrangements that are hardly possible without



Haskell: fib

Assignment text

The **Fibonacci numbers** $f_0, f_1, \ldots$ 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 Kate Milliken

submitted at Jun 03, 2007 06:32 AM, state: Submitted

Answer:

```
fib :: Integer -> Integer
fib n = case n of
        0 -> 0
        1 -> 1
        n -> (fib (n-1)) + (fib (n -1))
```

kate.20070603.073219.txt — Plain Text, 0Kb

Auto feedback:

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

  Expected result: 21
  Received result: 128
```

by Kate Milliken — last modified Mar 02, 2010 08:39 PM

Fig. 10. Example feedback from the Haskell backend.

such a technological basis. Thus, such a paradigm shift may be attractive for teachers in all study subjects that work with student assignments.

The benefits for the learning processes are even more substantial when electronic submission of students' assignments is coupled with automatic assessment. Automatic assessment allows for timely, almost immediate feedback for students, which is known to be an additional motivating factor.

The automatic testing of programming assignments was 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.

While the feedback provided by our backends (see Fig. 10) 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. 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 solutions. Thus, even though the automatic feedback may not yet be perfect, it represents a notable improvement for the students' learning experience.

A seemingly minor change in the organization and technical basis of exercises—i.e., introducing the constraint that all assignments and all students' solutions are electronic documents in a CMS—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.

The processes within the exercise courses have changed much more radically than initially envisaged, especially by the use of ECAssignmentBox and ECAutoAssessmentBox.

When we started using CAA and other e-learning components, we had the primary motivation of relieving teachers and students from administrative burden by automating certain processes and supporting others. Our experience is, however, that the change in the way that assignments are submitted has led 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.

The usage of ECAutoAssessmentBox in Rostock (see Section 5.2) on a weekly basis was the first broadscale usage of this CAA module at a site that is not the site of the developers. As we have learned, this experiment is judged as successful by both the teachers and the students in Rostock. This demonstrates as well the flexibility and the generality already realized and embodied in the architecture of the whole system.

The same applies for the usage of our automatic assessment approach at the LMU Munich, which developed and used an own backend for SML (see Section 5.3).

Nevertheless, every successful usage—especially from users completely independent from the original developers —is likely to create new demands.

The experiences in Rostock and Munich will lead to higher flexibility both in the testing and in the interaction with the students as well. In the backends based on test data, there is currently only one setup implemented: Testing succeeds only when all results from the student's solution agree with the corresponding results from the master solution and testing is stopped completely whenever there is a discrepancy between a student's result and an expected result. This discrepancy is then reported to the student and should help him to improve his solution.

Based on the suggestions from Munich and Rostock, the following alternative in the course setup, as well as in the feedback reporting will be realized and offered as alternative: Even when there is a discrepancy between a student's result and an expected result, testing will continue with the remaining test cases and test data. Reporting will be more informative by mentioning the ratio of successful to unsuccessful tests as well. Such a feedback may be more appropriate than the current one in cases where, e.g., a student has already covered the regular cases in his solution but merely has failed to treat a single special case. We expect that the flexibility gained will help to avoid frustration and strengthen motivation.

7 Conclusion

We have reported about the development of a flexible service-oriented system—ECSpooler and backends for different programming languages—for automatic assessment of programming assignment in CS education. We showed how this system can enable different frontends, i.e., learning management systems, to automatically assess programming assignments. We have also reported about experiences with using such a system in computer science education, both at our site and at other educational institutions.

In this service-oriented architecture, all common aspects of managing testable assignments (e.g., submission, storage, and result reporting) are encapsulated in the frontend. Only the specifics of the testing itself (e.g., for programming tasks: which programming language with which interpreter or compiler and with which test method) are realized as self-contained services—so-called backends. In conjunction with the spooler, backends offer a flexible and portable alternative to extend a learning management system or other e-learning environments with functionality for automatic testing of programming assignments. However, backends do not exist for programming languages only. They have also been developed and deployed for other formal notations that are amenable for automatic testing like regular expressions, XSLT transformations, or UIMA analysis engines. There have even been experiments with automatic support for the grading of assignments in natural language (cf. [28]), but this definitely needs more research work.

Since 2005, we have employed the eduComponents in all our lectures. Automatic testing is essential for all courses that have a focus on issues of algorithms and programming. But even with nontestable assignments, the electronic submission—and therefore, permanent storage, availability, and reusability as electronic documents—of students' solutions had an enormous impact on the teaching and learning arrangements in our courses. This has been illustrated above.

8 Outlook

In the meantime, a large collection of assignment tasks has been assembled. Now questions of how to make best use and reuse of these learning objects become pressing. We have experimented with an ontology-based organization of our repository and with respective search facilities (cf. [32]).

The submissions of students are stored together with log data (e.g., about number of attempts or about submission time and dates). This allows for data mining or at least data analysis with respect to patterns in students working behavior and for post hoc classification of the submissions. Since we know the exam results, we can try to correlate, e.g., the estimated average originality of a student's solutions or other possible indicators with the respective outcome in the exam. Originality may be estimated by measuring how distant or close a student's submitted program and documentation texts are compared to his peers' submissions for the same task.

We and—as the feedback shows (cf. Section 6)—our students are content with the new possibilities that the e-assessment based learning technology offer for teaching and learning and no one advocates a return to a conventional unautomated approach. But technology is just an enabling factor, the responsibility for success still lies with the people—educators, as well as students—making proper use of it.

Acknowledgments

The authors would like to thank their former colleague Dr. Michael Piotrowski for inspiring discussions and his contributions to the eduComponents, Wolfram Fenske and Sascha Peilicke for their substantial implementation work, and their colleagues Ilona Blümel and Dr. Manuela Kunze for their valuable feedback from their experiences in teaching with the eduComponents. The authors also thank the anonymous reviewers for their constructive criticism and their very detailed and valuable feedback. This paper is an extended and improved version based on prior publications [1] and [2] by the authors.

References

[1] M. Amelung, P. Forbrig, and D. Rösner, "Towards Generic and Flexible Web Services for E-Assessment," *Proc. 13th Ann. Conf. Innovation and Technology in Computer Science Education (ITiCSE '08),* pp. 219-224, 2008.

[2] M. Amelung, K. Krieger, and D. Rösner, "Flexibles E-Assessment auf Basis einer Service-orientierten Architektur," *Proc. Lernen im Digitalen Zeitalter (DeLFI '09): 7, E-Learning Fachtagung Informatik,* A. Schwill and N. Apostolopoulos, eds., pp. 247-258, 2009.

[3] R.J. Light, *Making the Most of College: Students Speak Their Minds.* Harvard Univ., 2001.

[4] M. Amelung, M. Piotrowski, and D. Rösner, "Educomponents: A Component-Based E-Learning Environment," *Proc. 12th Ann. SIGCSE Conf. Innovation and Technology in Computer Science Education (ITiCSE '07),* p. 352, 2007.

[5] M. Piotrowski, "Document-Oriented E-Learning Components," PhD thesis, Dept. of Computer Science, Otto von Guericke Univ., 2009.

[6] D. Rösner, M. Piotrowski, and M. Amelung, "A Sustainable Learning Environment Based on an Open Source Content Management System," *Proc. German E-Science Conf. (GES '07),* 2007.

[7] M. Piotrowski, M. Amelung, and D. Rösner, "Tactical, Document-Oriented E-Learning Components," *Proc. IADIS Int'l Conf. E-Learning,* pp. 171-177, 2007.

[8] P. Black and D. Wiliam, "Inside the Black Box: Raising Standards through Classroom Assessment," *Phi Delta Kappan,* vol. 80, no. 2, pp. 139-148, 1998.

[9] K.M. Ala-Mutka, "A Survey of Automated Assessment Approaches for Programming Assignments," *J. Computer Science Education,* vol. 15, no. 2, pp. 83-102, June 2005.

[10] G.E. Forsythe and N. Wirth, "Automatic Grading Programs," *Comm. ACM,* vol. 8, no. 5, pp. 275-278, 1965.

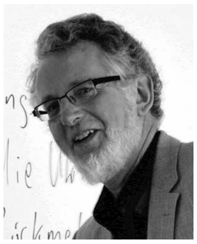
[11] M. Laakso, T. Salakoski, A. Korhonen, and L. Malmi, "Automatic Assessment of Exercises for Algorithms and Data Structures—A Case Study with TRAKLA2," *Proc. Fourth Finnish/Baltic Sea Conf. Computer Science Education,* pp. 28-36, Oct. 2004.

[12] R. Saikkonen, L. Malmi, and A. Korhonen, "Fully Automatic Assessment of Programming Exercises," *Proc. Sixth Ann. Conf. Innovation and Technology in Computer Science Education (ITiCSE '01),* pp. 133-136, 2001.

[13] M.T. Helmick, “Interface-Based Programming Assignments and Automatic Grading of Java Programs,” *Proc. 12th Ann. SIGCSE Conf. Innovation and Technology in Computer Science Education (ITiCSE ’07),* pp. 63-67, 2007.
[14] M. Striewe, M. Balz, and M. Goedicke, “A Flexible and Modular Software Architecture for Computer Aided Assessments and Automated Marking,” *Proc. First Int’l Conf. Computer Supported Education,* pp. 54-61, 2009.
[15] C.A. Higgins, G. Gray, P. Symeonidis, and A. Tsintsifas, “Automated Assessment and Experiences of Teaching Programming,” *J. Educational Resources in Computing,* vol. 5, no. 3, p. 5, 2005.
[16] M. Joy, N. Griffiths, and R. Boyatt, “The BOSS Online Submission and Assessment System,” *J. Educational Resources in Computing,* vol. 5, no. 3, p. 2, 2005.
[17] C. Beierle, M. Kulaš, and M. Widera, “Automatic Analysis of Programming Assignments,” *Proc. der 1. E-Learning Fachtagung Informatik (DeLFI ’03),* A. Bode, J. Desel, S. Ratmayer, and M. Wessner, eds., vol. P-37, pp. 144-153, 2003.
[18] J. Krinke, M. Störzer, and A. Zeller, “Web-Basierte Programmierpraktika Mit Praktomat,” *Proc. des Workshop Neue Medien in der Informatik-Lehre,* Oct. 2002.
[19] O. Gotel, C. Scharff, and A. Wildenberg, “Teaching Software Quality Assurance by Encouraging Student Contributions to an Open Source Web-Based System for the Assessment of Programming Assignments,” *Proc. 13th Ann. Conf. Innovation and Technology in Computer Science Education (ITiCSE ’08),* pp. 214-218, 2008.
[20] S.H. Edwards and M. Peréz-Quiñones, “Web-CAT: Automatically Grading Programming Assignments,” *Proc. 13th Ann. SIGCSE Conf. Innovation and Technology in Computer Science Education (ITiCSE ’08),* 2008.
[21] A. Hoffmann, A. Quast, and R. Wismüller, “Online-Übungssystem für die Programmierausbildung zur Einführung in die Informatik,” *Proc. Die 6. E-Learning Fachtagung Informatik (DeLFI ’08),* 2008.
[22] M. Rahn and J. Waldmann, “The Leipzig Autotool System for Grading Student Homework,” *Proc. Workshop E-Learning,* M. Hanus, S. Krishnamurthi, and S. Thompson, eds., 2002.
[23] Epaile, Automatic Grading of Programs, http://docs.moodle.org/en/Student_projects/Automated_grading_of_programs, 2010.
[24] N. Bieberstein, S. Bose, M. Fiammante, K. Jones, and R. Shah, *Service-Oriented Architecture (SOA) Compass: Business Value, Planning, and Enterprise Roadmap.* IBM, 2005.
[25] M. Bell, *SOA Modeling Patterns for Service Oriented Discovery and Analysis.* Wiley, 2010.
[26] G.J. Myers, *The Art of Software Testing,* second ed. Wiley, 2004.
[27] N. Provos, “Improving Host Security with System Call Policies,” *Proc. 12th USENIX Security Symp.,* Aug. 2003.
[28] T. Feustel, “Analyse von Texteingaben in einem CAA-Werkzeug zur Elektronischen Einreichung und Auswertung von Aufgaben,” Master’s thesis, Dept. of Computer Science, Otto von Guericke Univ., 2006.
[29] R. Binder, *Testing Object-Oriented Systems: Models, Patterns and Tools,* J.C. Shanklin, ed. Addison-Wesley, 2000.
[30] K. Claessen and J. Hughes, “QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs,” *Proc. Fifth ACM SIGPLAN Int’l Conf. Functional Programming (ICFP ’00),* pp. 268-279, 2000.
[31] A. Zeller, “Making Students Read and Review Code,” *Proc. Fifth Ann. SIGCSE/SIGCUE ITiCSE Conf. Innovation and Technology in Computer Science Education (ITiCSE ’00),* pp. 89-92, July 2000.
[32] M. Otto, “Ontologien zur semantischen Suche in einem Bestand von Dokumenten,” Master’s thesis, Dept. of Computer Science, Otto von Guericke Univ., 2008.

Mario Amelung studied business informatics from Otto-von-Guericke University (OvGU), Magdeburg, Germany. Since 2005, he has been working toward the PhD degree at WDOK. He is a member of the board and senior project manager at Eudemonia Solutions AG, where he works on different software engineering projects. His academic work focuses on service-oriented architectures and e-assessment in computer science education.

Katrin Krieger received the master’s degree in computer science and education science from Otto-von-Guericke University (OvGU), Magdeburg, Germany, in 2006. She was awarded a PhD scholarship from the Hasso-Plattner Institute and spent 18 months in Potsdam, Germany. Since 2009, she has been a scientific assistant in the Department of Computer Science at OvGU and works in the Knowledge-Based Systems and Document Processing Research Group. Her research interests include e-learning, recommendation engines, web technologies, and collective intelligence.



Dietmar Rösner has been a full professor in the Department of Computer Science at Otto-von-Guericke University, Magdeburg, Germany, since 1995. He is the head of the Knowledge-Based Systems and Document Processing Research Group. His research interests include natural language processing, XML technologies, emotions in human-machine interaction, and e-learning.