Towards Electronic Exams in Undergraduate Engineering

A Project on Numerical Mathematics in eExams at the University of Stuttgart

Thomas Richter, David Boehringer
IZUS/TIK Computing Center
Stuttgart, Germany
{richter,boehringer}@tik.uni-stuttgart.de

Abstract—In this work, we describe the first step towards electronic exams at the University of Stuttgart. The motivation, both driven by students and lecturers, is to offer assessments that are closer to real-life programming by allowing interactive programming and debugging not possible with pen and paper exams. This is in so far important as skills in computer algebra systems as for example MATLAB or programming languages like Java are an elementary ingredient of any engineering study nowadays.

In this paper, we describe the technological basis for computer based exams based on a system already successfully deployed for homework assignments, and describe its transition to an electronic assessment system. We discuss not only the technological requirements on software, its architecture and our choices for hardware, but also provide a brief introduction into the organizational and legal challenges that we are going to overcome.

Virtual Laboratory; eLearning Management System; eExam

I. INTRODUCTION

Numerical mathematics is an essential topic in all undergraduate engineering courses; students not only learn mathematical tools and mathematical algorithms to discuss real-life problems by means of a computer, they also learn fundamentals of numerical tools such as MATLAB or Octave, and principles of software development in languages like Java or python. As demanded by the eBologna process, frequent tests and assessments are required to continuously monitor and evaluate the learning success of our students; needless to say, it is very attractive to conduct exams in fields that require any type of computer programming on computer systems: algorithms in the form of computer programs can be verified immediately, students get feedback immediately as well and can eliminate typos and oversights within seconds. Lecturers and teaching staff will retrieve syntactically correct source code, and can run a quick correctness test of the handed in solutions by testing against example data; students are assessed in an environment that mirrors more realistically the scientific workplace.

Even though we already implement electronic homework assignments, exams are – as of today – pen and paper tests, even in fields closely related to computer science. Reasons for this are manifold: first, legal requirements enforce to document and archive exams over a long period of time, typically ten years. Any document must be archived in a shape admissible by law, where current legal regulations only cover paper-based tests. Second, study and examination regulations did not mention nor allow any other form of exam until recent changes were made. Third, while our software infrastructure is capable enough to handle hundreds of students simultaneously, we lack sufficient computer workplaces to accommodate undergraduate courses.

This paper describes a three-year project funded by the University of Stuttgart to implement electronic exams in undergraduate engineering courses. It is based on a well-tested and established "virtual programming lab" described in detail in [1]. We also describe its integration into the eLearning infrastructure, and our plans how to address the shortcomings of the technological infrastructure. We also shortly discuss how to overcome the legal predicaments of eExams.

II. TECHNOLOGICAL BASIS

The computer tools of choice for teaching undergraduate mathematics and undergraduate engineering are MATLAB and C++ at our institution. Elementary programming courses for freshmen in the transition from school to university use kava to establish a first contact to a programming language. While our programming lab also offers support for Octave, DUMUX and C, these tools are less prominent in undergraduate education.

As described in [1], we developed in the last years a virtual programming lab (ViPLab) that allows students to conduct homework exercises from their own computer remotely while requiring a web-browser only. Student code is edited in a java applet, and code entered there is transmitted over the internet for server-side compilation and evaluation. Figure 1 shows a typical screenshot of the student view on our programming lab, both with the student input on the left and the program output on the right. Output can be both textual and graphical, and includes compiler error or warning messages if any problems
were found. The tool also allows students to jump directly to compiler errors and includes elementary editor facilities such as syntax highlighting and line indentation. Source code to be edited is typically constrained to a single source file, but support for multiple sources, including header files, were added in later releases due to popular demand. Note, however, that ViPLab does not aim at replacing integrated desktop environments (IDEs) such as Eclipse or Visual Studio, and hence offers a constraint and cut-down editor restricted to elementary functions. It is disputable whether handling of IDEs should be in focus of undergraduate courses as it might distract students from learning a computer language.

Once an exercise is ready and successfully tested, it gets both uploaded to a server, and downloaded to the teacher’s computer in the form of a SCORM archive. This SCORM archive, essentially a ZIP file containing a html page and a link to the student applet plus a link to the exercise, then needs to be manually uploaded to the LMS.

The student front-end comes currently in the form of a java applet that is deployed in the form of SCORM [2] packages. These packages then become integral components of homework assignments on the Learning Management System (LMS). Students thus get access to the system by means of the eLearning infrastructure of the university and hence rarely notice the difference between the LMS and ViPLab. The eLearning infrastructure which delivers such packages in Stuttgart is ILIAS [3], though any other SCORM compliant system, such as moodle [4], would be suitable as well.

Teachers use a less restricted front-end to create ViPLab exercises. Even though it also comes in the form of an applet, it allows adding or removing source files, or blocks of source code. Such blocks can be marked as editable, constant or invisible, were the latter allows for integration of library functions allowing graphical output, or invisible test code to feed the student code with sample data. Figure 2 shows a screenshot of the lecturer tool, here on a more complex programming exercise requiring more than one source file.

III. RELATED WORK

Quite similar to ViPLab, the University of Sydney offers the GROK system [4] for facilitating learning of computer programming languages. GROK is, very similar to ViPLab, a client-server architecture with students accessing servers indirectly through a graphical client front-end run in the browser. GROK also offers some limited form of interactivity in programs, i.e. programs may stop and request user input; this is currently not yet available in ViPLab, mostly due to lack of demand from lecturers, though we planned at a time to include a teacher-configurable graphical user interface that would allow students to enter data for their programs. The typical size and complexity of student programs did, however, not yet require this extension.

Other LMS specific solutions exist; for example, the VPL module [13] for moodle provides a comparable programming lab that integrates deeply into the LMS and hence does not provide a generic solution. Professional solutions like Maple-TA [14] requiring dedicated servers and proprietary software are also available on the market, though the latter focuses less on programming, but rather uses the Maple software as a tool to extend assessments with tests requiring mathematical algebra or higher mathematics for evaluation.

As pointed out in the introduction, our aim is the realization of electronic assessments: While the technical infrastructure is already present, we currently face three types of challenges: Technological challenges due to shortcomings in the current software architecture we can solve by advancing ViPLab alone; organizational challenges due to circumstances and limitations of the infrastructure of the university; and legal obstructions due to requirements under which assessments have to be conducted and managed. We will go into these cases now one by another.
IV. OVERCOMING TECHNOLOGICAL CHALLENGES

While the ViPLab system already runs in the eLearning Management System (LMS) of the university, technological restrictions of the infrastructure made it up to now impossible to feed student code back into the LMS and homework results. Students developed code within ViPLab, but the final solutions need to be submitted by email or by copying the code manually into the clipboard and inserting it into a text field in the LMS. Reasons for this are due to limitations of the SCORM interface which only offers very limited means to store data from within a SCORM module (a so-called SCO) in the LMS. This is because the major design goal of SCORM is content-delivery, and not the establishment of an integrative software architecture. Even though SCORM does offer some limited means to inject data into the LMS, we found that the ILIAS system does not provide its users – the teachers in our case – access to such data, mainly due to a software defect.

Needless to say, it would be – however – very desirable to avoid the round-trip through the clipboard for submitting solutions, and to enable teachers to test and inspect student code directly within the LMS. We thus checked alternative standards other than SCORM for their applicability for our use-case: The LTI standard is a lightweight alternative that focuses mainly on assessments. LTI triggers from within an LMS external assessment tools, and collects, in its first version, only a single numerical score from them. While this is sufficient for elementary automatized assessment tools, it is unsuitable for complex exercises that require manual evaluation and thus storage of the complete student answer. The follow-up standard, LTI 2.0, provides a richer interface, but is currently still in a draft status and not widely supported by LMSs.

We thus decided to leave the route of standards and standard-defined interfaces and are instead developing an ILIAS specific solution that offers a deeper integration of ViPLab into the LMS. While the details of the software infrastructure will be reported elsewhere [6], a short overview shall be given here:

ViPLab, as of today, uses a three-tiered design: The front-end in the form of a java applet embedded in an html page and delivered as a SCORM package; several numerical back-ends compiling, linking and running student code in "choot-cages", and a middle-ware between the two system components that provides load-balancing facilities and also keeps the exercise database. Communication between the system components is based on the REST architecture [7]. Figure 3 provides an overview on the existing architecture.

The extended architecture will now establish a new communication channel between Front-End, based on JavaScript, and the LMS. This interface will collect the student solution via the JavaScript-Java bridge, and feed it back into the LMS via a hidden field on the html page delivering the exercise. A new dedicated ILIAS test type will collect the solutions, and to enable teachers to test and inspect student code directly within the LMS.

V. ORGANIZATIONAL CHALLENGES

As long as ViPLab is used for homework assignments, students are of course encouraged to use their own machines from home to access the LMS, and thus run the ViPLab exercises within the LMS. This model is, of course, no longer applicable for exams where internet access must be limited and the execution environment must be restricted to ensure fairness and prevent cheating. University maintained computer pools are the ideal choice to address these requirements: the software installation and network availability is fully within the control of the university, and access to external devices can be prevented by disconnecting or disabling any type of external interface connector, let it be USB, eSATA or firewire.

Unfortunately, the available computer pools at the campus of the University of Stuttgart are too small and do not offer the necessary capacity for typical undergraduate courses – which we estimate to be at least 400 students per exam. Several solutions are possible to address this predicament:

In one possible option, a single assessment could be split into multiple sequential tracks, limiting the number of students per track to the capacity of the pools. The disadvantage of this solution is that one cannot completely prevent any type of communication between students, which can be addressed by varying the exercises – which again requires the creation of multiple, but comparable tests.

A second possible option is to equip students with mobile hardware and organize assessments within regular lecture halls depending entirely on wireless internet access. The mobile devices could be handed out upfront, in exchange for an ID, and would be returned at the end of the test. Other than that, requirements similar to those for computer pools hold.

We currently consider the second solution as the more viable option; though note that this use-case enforces certain requirements on the choice of the hardware which we will go into in the following paragraphs.
It is important that we can control which network the systems connect to, which applications students can start, and how easily a large amount of identical hardware can be configured or installed simultaneously. The running time of a suitable mobile device also needs to be sufficient to last for the duration of an exam, and the device should be relatively inexpensive, should survive rough handling by students and should be easy to replace. A keyboard would be essential for programming exercises. The physical dimensions of the system should be large enough to allow readable font sizes, but small enough to fit on the small desks in lecture halls. Internal storage is less critical as systems only need to boot a minimal environment and solutions are stored in the LMS anyhow.

We checked several options for hardware, amongst them regular notebooks, netbooks with remote-managing software, in particular Citrix XenDesktop, Tablets and Google ChromeBooks. Results of this evaluation are depicted in Table 1.

Laptops represent here the large variety of standard PC compatible hardware. Netbooks represent dedicated hardware designed primarily to work on the internet. They typically come with restricted internal storage, and run applications within browsers. Multiple solutions exist; we tested in particular netbooks based on Citrix XenDesktop [12] and google ChromeBooks.

<table>
<thead>
<tr>
<th></th>
<th>Laptops with standard Os</th>
<th>Netbooks &amp; Citrix</th>
<th>Chrome Books</th>
<th>Tablets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Costs</td>
<td>-</td>
<td></td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Remote Management</td>
<td>--</td>
<td>+</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Access Control</td>
<td>--</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Handling</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>--</td>
</tr>
<tr>
<td>Battery Time</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Runs ViPLab</td>
<td>++</td>
<td>+</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Evaluation matrix of required features vs. choices of hardware, as found by our evaluation. See text for the criteria.

XenDesktop runs standard PC applications via a remote-desktop protocol from an application server, whereas chrome books require HTML5 based web-applications that run natively in a browser. Google offers similar to Citrix a very rich remote management tool that allows blocking functionality of the device, in particular to block access to IP addresses.

While chrome books typically come with a keyboard, tablets represent touch-screen devices lacking it and depending entirely on touch-screen or pen input. Other than that, applications come here either as dedicated apps, or in the form of browser-based solutions. This class is very diverse and availability of apps depends on the particular vendor.

In Table 1, “costs” are the estimated costs for the solution, where an empty column indicates that it depends on the particular choice and no general answer can be given.

“Remote management” indicates the ability of the solution to be setup and managed remotely, in particular to configure large sets of identical hardware in the same way. Scalability is given with Citrix and Chrome books, the latter through the proprietary Google management tool. For Laptops, a suitable tool would need to be implemented.

“Access control” indicates the amount of control a remote management has on the overall solution, i.e. the ability to control and block access to internet resources or hardware ports. The less open connectors a device offers the better.

“Handling” provides the overall usability of the device in electronic exams. Tablets lack a keyboard, which may already render them unusable for programming classes.

“Battery time” is the estimated running time of the device with a fully charged battery. It often depends on the particular device, though Tablets show here a clear advantage.

Finally, to run ViPLab in its current form, the device would need to run Java applets.

As seen from Table 1, there is no ideal solution at this time. Laptops are capable enough to address all our needs in terms of pure hardware requirements, but additional software needs to be installed to allow remote management of huge device classes, for example the Citrix software. This device class is typically the most expensive choice, as it also includes many hardware features we would not need and would even need to turn off, e.g. external USB connectors.

Netbooks with remote management and remote desktop solutions are entirely sufficient, though fewer models are available. We found, however, that running remote services on Netbooks is less than ideal and partially cumbersome, and google Chrome Books do not run Java natively. Tablets lack a keyboard which makes them unsuitable for any programming work.

Despite hardware choices, however, one important outcome of this study is that Java and Java applets are more and more becoming a show-stopper. We hence decided not only to implement a new interface towards the LMS, but also to re-implement the front-end in javascript, enabling us to run it on any type of device. While we have not yet made a final decision, many arguments speak for Google Chrome books. However, privacy concerns from students may prevent this from happening given that the device management software is hosted by Google. Even though private user data is never stored on the devices or on Google accounts, recent developments increased the public awareness of the matter and may cause irritations from the student side.

VI. LEGAL OBSTRUCTIONS

Electronic homework exercises caused little or no problems from a legal perspective. While they would require a deeper integration into the curriculum from a legal perspective for some courses, we found a way to work around this issue whenever it appears: electronic homework exercises allow students to collect extra credits which can be used as a bonus to compensate the outcome of the final exam.

For exams, however, stricter legal requirements are in force: they must be conducted in a controlled environment ensuring the identity of the examinee, and ensuring that the
The examinee is admitted to participate; the solutions handed in by students must be archived in a way admissible by law over a period of several years, where details depend on the state; the university must allow students to inspect their solutions later on, and allow them to review the evaluation; examination is required to be fair, i.e. conditions must be similar for all participants, and cheating must be prevented, or at least impeded as much as possible; technical conditions must be satisfactory allowing a smooth execution of the assessment. Finally, the assessment needs to conform to the examination conditions regulated by the corresponding department. Such regulations define, for example, whether the exam is an oral or a written assessment, though electronic assessments were, until lately, typically not included.

Currently, the requirements are addressed by checking the ID card of students during the exam, by keeping the original in paper form, and by providing students access to the corrected and evaluated solutions at an appointed review time. Fairness is usually addressed by providing either identical or similar exercises for all participants, whereas cheating is impeded by strict supervision in a lecture hall.

As far as the overall legal situation in Germany is concerned, electronic exams have been organized and conducted at several German universities, but whether this form of assessment is valid and admissible by law has, to our knowledge, never been tested in court. Thus, at this point, we can only try to minimize the legal risk, but we cannot completely avoid it.

In a first step, a necessary legal foundation was laid by changing the study and examination regulations at the participating departments by allowing eExams as a form of assessment. Furthermore, electronic exams require the acceptance of electronic exams by the registrar’s office of the university. Fairness can be granted by similar constructions as in pen and paper assessments, whereas impediment of cheating can be addressed both by supervision as in traditional exams, as also by technical means by blocking access to external devices or network resources.

Our plan is to run the exam on a dedicated server, separated from the regular LMS available only during the exam, and not accessible from outside the university. Students will log into this server using their LMS login, to be validated manually by teaching staff during the exam, and activity on this server will be tracked and logged. This will, as we hope, provide enough proof to ensure the students’ identities, and makes student activity transparent. Once completed, corrected and evaluated, the logs of student activity and the handed-in exercises will be printed out, and hence will be archived in the traditional way on paper allowing review and inspection. Despite the actual student solution, we would also need to ensure that the exam run smoothly from a technical perspective and students were not hindered by complications of the infrastructure or software. Thus, it is necessary to log student activity on the eExam system and archive these logs along with the handed-in solutions.

VII. PEDAGOGY AND EVALUATION

While electronic homework exercises with ViPlab are already accepted by students and lecturers, we were curious in how far they were effective. At this time, we cannot provide an answer for electronic exams as they have not yet been implemented, but a partial answer can be given in how far ViPlab is a suitable tool for homework assignments.

For that, we conducted a study comparing the learning achievement through ViPlab with that of a programming course based on traditional software tools such as a restricted IDE and a command line. Comparison with pen-and-paper homework assignments are less meaningful as programming classes have been performed on computer systems for quite a while; the difference is here that for them students had up to now to use computer pools of the university, or had to install compiler and development environment on their own systems.

We thus conducted a study on a two-week intensive course, where one group of students used ViPlab to do their homework exercises, while a control group worked with a traditional setup consisting of a restricted editor and a console running on Linux machines.

<table>
<thead>
<tr>
<th>N</th>
<th>Control Group</th>
<th>Treatment Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score</td>
<td>1.88(0.86)</td>
<td>2.04(0.65)</td>
</tr>
<tr>
<td>TECOW1</td>
<td>0.50(0.22)</td>
<td>0.62(0.21)</td>
</tr>
<tr>
<td>PRACOW1</td>
<td>0.59(0.22)</td>
<td>0.66(0.19)</td>
</tr>
<tr>
<td>COMA</td>
<td>2.52(1.03)</td>
<td>2.18(0.76)</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>2.76(1.03)</td>
<td>3.12(0.61)</td>
</tr>
</tbody>
</table>

Table 2: Outcome of the comparative ViPlab study, comparing ViPlab with a traditional editor/console setup on Linux. Left: Control group, Right: Treatment group. TECOW1 and PRACOW1 are theoretical and practical computer knowledge scores as defined by the INCOSBI-R test [9],[10]. COMA measures the student emotional assurance for using computers from the same test tool. TECOW1 and PRACOW1 range from zero to one, COMA from one to three. “Score” is the mean achievement of our assessment and measures the learning outcome. “Satisfaction” is the mean opinion score of students on the corresponding development environment, based on questions from [11]. Even though differences seem to be present, they are not statistically significant.

The outcome of this test was first reported in [8] and shall be briefly recapitulated here, too: The most important result was that we found no statistically significant difference in the achievements of the learning goal between the traditional setup and ViPlab. The only difference we found was the acceptance rate of the two models comparing students that had a prior knowledge in using the Windows operating system with those that worked with Apple or Linux. While students with Windows experience felt less comfortable with the traditional editor and console setup running on Linux, no statistically significant differences between Windows and Apple/Linux users were found for ViPlab. Both groups, the study group and the control group, felt equally comfortable with the web-based ViPlab client. Overall results, independent of the individual student background, are reproduced in Table 2.

Overall, one can conclude that ViPlab works as well as a traditional IDE installation, though lecturers need less time to prepare exercises in ViPlab than with a traditional setup. The latter requires manual preparation of program segments; the former supports the lecturer with an interactive graphical tool that allows them to compile exercises within a single environment. Some sophisticated tools that are hard to install and maintain, e.g. DUMUX, would not be available for homework assignments without ViPlab.
VIII. CONCLUSION AND OUTLOOK

Electronic exams, especially in fields testing computer-based skills, are desirable since they allow a more realistic and more meaningful assessment technique than traditional pen and paper tests. As our study shows, restricted web-based development tools can replace traditional IDEs without compromising the learning achievement or the student satisfaction; as added benefit, they enable students to use their own computer systems for homework assignments without requiring troublesome installation.

However, not only organizational and technological challenges prevent a quick transition to electronic exams; legal demands are actually much harder to address, while the former only require the choice of a suitable technology and a suitable infrastructure.

IX. ACKNOWLEDGEMENTS

The authors want to thank the IZUS/TK Computing Center and the University of Stuttgart for funding this work.